今期內容：
- 信用賬項處理系統
- CP/M檔案加上密碼保護
- DISKSCAN軟/硬碟機檢查程式
- 高速MODEM新標準CCITT V.32

微電腦技術
Compu-Link

1
1986 VOL.3
港幣十元

NETWORKING & BUSINESS APPLICATION JOURNAL



PHC-16
PLASMA EFFECT XT PORTABLE

總代理：
VINCENT COMPUTER CENTRE LTD.
雲生電腦有限公司
九龍尖沙咀柯士甸路22-26號好兆年行608室 電話：3-7211288-0 電訊：HX 38316 VCC HK

IMPACT DOT MATRIX PRINTER

Super 5



EP-1802 EP-1201 EP-1000
— OPTIONAL RS232C & 8K BUFFER
— OPTION CARD FOR APPLE™ IIC & MACINTOSH**



EP-1802

- 180 CPS (TEXT) 33 CPS (NLQ) 112 CPS (PROP)
- FIVE MODES OF TEXT PRINTING SELECTABLE ON FRONT PANEL
- 7K BUFFER EXPANDABLE TO 11 KB
- HEX DUMP MODE FOR PROGRAMMERS' CONVENIENCE

EP-1201

- 120 CPS (TEXT) 29 CPS (NLQ) 75 CPS (COMP/PROP)
- NEAR LETTER QUALITY (NLQ)
- 80/132 COLUMN EASILY SWITCHIBLE
- SUPPORTS IBM* PC MATRIX MODE PLUS TWO GRAPHICS MODE
- 1K BUFFER EXPANDABLE TO 5KB

A & N ENTERPRISE CO., LTD.
20/F., Western Centre,
40-50 Des Voeux Road, West
Hong Kong
Phone:
Fax: 5-4
Telex:

HONG KONG
MICROGRA
Shop No.
150 Fuk Wa
Tel: 3-7280

OCEAN MICRO
Basement Shop
150 Fuk Wa
Tel: 3-884231
Telex: 52269





EP-1000

- 100 CPS (TEXT) 20 CPS (NLQ) 86 CPS (COMP)
- NEAR LETTER QUALITY (NLQ)
- 80/132 COLUMN EASILY SWITCHIBLE
- WORD PROCESSING FEATURES AUTO CENTERING / AUTO JUSTIFICATION LEFT ALIGNMENT / RIGHT ALIGNMENT
- PRINT MODE SELECTOR SWITCH STD.PRG./NLQ/COMP.



EP-2405

- HIGH SPEED PRINTING AT 240 CPS IN DRAFT, 51 CPS IN NLQ
- 136 COLUMN WITH BI-DIRECTIONAL PRINTING AND LOGIC SEEKING CAPABILITIES
- FULL CHARACTER PITCH FEATURE (CPI): DRAFT/NLQ (10, 12, 15, 17 AND PS)
- MULTI NLQ FONTS (COURIER, BOLD PS)
- CENTRONIC PARALLEL AND SERIAL INTERFACE ARE BOTH STANDARD
- 15K BUFFER

EP-18

- HIGH SPEED PRINTING AT 180 CPS IN DRAFT, 38 CPS IN NLQ
- 136 COLUMN WITH BI-DIRECTIONAL PRINTING AND LOGIC SEEKING CAPABILITIES
- FULL CHARACTER PITCH FEATURE (CPI): NEAR LETTER QUALITY MODE (10, 12, 15, 17 AND PROPORTIONAL SPACING (PS) NLQ, ETC.)
- AUTO PAPER LOADING FOR CUT SHEET
- OPTIONAL AUTO CUT SHEET FEEDER & RS232C SERIAL

EP-2405 EP-1805
— PAPER WIDTH 4-16.5 inches (102-394mm)
— OPTIONAL BUFFER CHIP

* IBM is a registered trademark of International Business Machines Corp.
** Apple is a registered trademark of Apple Computer, Inc.
Macintosh is a trademark licensed to Apple Computer, Inc.
* ESPON is a trademark of Epson Corp.
* Diablo is a registered trademark of XEROX Corp.

香港教育司署立案（1957年創校）

世界函授學校

規模大　優點多　科目好

導師
國內外著名大學學士，經香港教育司考核合格，教導認真，決不馬虎。

學習時間
函授時間是沒有時間規限，如時間充裕可提早完成所學的科目，提前畢業。

文憑
本校頒發文憑，各大機構認許，實是最佳學歷證件。

不收雜費
學生只需要交低廉之學費外，所有課本，試題及測驗紙張均免費派發。

校址： 香港九龍彌敦道五六六號9樓A座
電話：③—803532

初級電腦科
電腦基本原理，蘋果牌，IBM,PDP等等電腦操作，程式編寫，八個月畢業。

初級無綫電科
無綫電基本原理，收音機製作及修理，全部圖解教導，十個月畢業。

半導體裝修科
講解最新半導體IC集成電路及FM身歷聲收音機擴音機原理，十個月畢業。

實用數理
可以加强數理基礎，講授幾何、代數、三角及計數機運用，六個月畢業。

x+y=z

實用商科
講解經濟學識、銀行學、進出口貿易、商業管理，學習易，六個月畢業。

無綫電工程
講解無綫電綫路設計發射機及天綫設計，是深造的理想科目，一年半畢業。

汽車工程
最新汽車構造，內燃機原理，自動波風油軚動力剎制等原理，一年半畢業。

冷氣工程
空氣調節原理，設計及檢修，冷藏及製冰原理，冷氣機檢修，一年半畢業。

商業美術科
商業美術基本理論和技術，廣告設計，黑白畫圖案畫技術，十二個月畢業。

應用漫畫
繪畫基本技術、人物動作及表情畫法，活動卡通，世界漫畫，六個月畢業。

本港學生可利用電話詢問索取章程　電話③－803532

請依綫剪下，塡好姓名，回郵地址，貼在信封上投寄，本校即免費寄上簡章

郵件請寄：P.O.BOX 70011, KOWLOON, CENTRAL POST OFFICE
九龍中央郵政局信箱 70011 號

技術就是力量

（請註明要學的科目）
- ☐初級電腦科
- ☐無綫電工程科
- ☐初級無綫電科
- ☐半導體裝修科
- ☐實用數理科
- ☐汽車工程
- ☐冷氣工程
- ☐實用商科
- ☐商業美術
- ☐應用漫畫

請貼上郵票

香港教育司署立案

To：世界函授學校註册處

P. O. BOX No. 70011
KOWLOON CENTRAL POST OFFICE,
HONG KONG.

香港九龍中央郵局信箱70011號

（寄自）
姓名：
地址：

電話：K-803532

學問就是財富

微電腦技術 Compu-Link

VOL 3／NO.1　一月號　總第13期

一九八七年一月一日出版

出　　版：電腦時代出版社
編　　輯：微電腦技術編輯部
軟件支援：COMPUHOUSE
通訊地址：九龍中央郵政信箱71193號
通訊電話：3-7712007
讀者服務部：旺角上海街438號
同珍商業中心1104室
承 印 者：永恒印刷公司
發 行 人：同德書報社

程式磁碟特約經銷處：
地鐵及九廣鐵路沿線香港電視服務站
灣仔世紀電腦軟件公司
萬達電腦公司

COMPUNET 資訊網絡中心

特稿

36 你是否需要一部AT?

當 IBM PC/XT 的熱潮過去，現時最新的焦點便是P-C/AT，而且一般 AT 兼容機已降至普通用戶能負擔的水平，故請本文的作者帶大家認識 XT 與 AT 的分別，以及軟件兼容的問題。

33 免費贈送精美座枱年曆咭

34 新讀者需知

市場動態

12 港大安裝阿波羅「領域」3000工作站
改裝收銀機成終端機

新產品快訊

13 使用80386的AT 加速咭

54 簡易資料庫管理系統RAPIDFILE

51 最新擴展咭將PC 變爲圖文傳眞機

MODEM／TELECOM 專欄

14 高速MODEM 新標準CCITT V.32

高速 MODEM 的技術日益進步，現時最新的標準是C-CITT V.32，傳送速度爲 9600BPS。究竟這個標準的影響有幾多？本文的作者將爲大家剖析。

16 通訊軟件加上ANSI 色彩介面

dBASE/WORDSTAR應用

24 信用賬項處理系統(一)

雖然我們已介紹了一些 dBASE工商業應用程式，但要構成一個高效能的系統必須有更多的功能配合，本文的信用賬項處理系統能爲各商戶解決賬項不同過欵期的問題。

27 拯救WORDSTAR 失落的資料

CP/M 系統研究

18 CP/M 檔案加上密碼保護

CP/M-80（包括 APPLE CP/M）是一個開放性的操作系統，並沒有考慮保安的問題。本文的程式能替可執行檔加上密碼保護，使大家無後顧之憂。

20 爲TURBO PASCAL 增加音樂及說話功能

一連數期陳彥君都爲大家介紹了 APPLE 版本 TURB-O PASCAL 的畫面應用技巧，今期則是音樂及說話方面的擴展，使 TURBO PASCAL 更加多姿多采。

1-2-3／SYMPHONY 專欄

28 專業資料分析系統：WHAT'S BEST

高級應用軟件評介

41 CONCURRENT PC-DOS

47 多用戶資料庫管理系統：PROGRESS

一家較具規模的公司通常每個部門都有獨立的微電腦系統，這時資料的分享便成爲難題，亦往往影響了工作效率。解決的方法是區域網絡系統與多用戶資料庫系統，本文所介紹的 PROGRESS 便是一個好例子。

IBM 系統研究

46 PC-DOS 3.2有BUG?

50 商用軟件磁碟保護技巧淺談

51 趣味遊戲選之蘋果棋

52 動態記憶管理技巧

55 DISKSCAN 磁碟機維護系統

軟／硬碟機的用戶常常遇到一個難題，就是軟／硬磁碟的可靠情度有幾多？假如存取一個會計紀錄時出現問題，後果不想而知。本文程式能爲大家徹底測試軟／硬磁碟，比D-OS 附隨的 CHKDISK 更加可靠。

超中文1987新突破！群

第一張完美24×24中文卡面世

＊超羣LE-2401,LE-2411系列中文卡＊

型號 ＼ 功能	適用主機PC, XT, AT或COMPATIBLE	24×24字形顯示	無需附加HERCULES GRAPHIC CARD	輸入方法：倉頡，簡易，注音，內碼	片語，筆劃，符號，電報，拼音	繁體字庫，不佔主機MEMORY	解像度	造字功能	片語功能	ON LINE HELP SCREEN	軟件保護功能	按鍵式純英文及中文狀態轉換	按鍵式內碼轉換 BIG5, 5550, 超羣，大千	NETWORK SUPPORT: ①SPERRY USERNET	②TRANSNET	③ETHERNET	④PCNET	⑤MSNET	⑥NOVELL	定價	附軟件：文書處理	其他軟件：PE2, 123, WS2, d3＋	訓練課程(免費)
超羣LE-2401	●	●	●	●	●	13053	960·700	826	1000 W	●	●	●	●	●	●	●	●	●	●	HK$5000	●	●	●
倉頡2025B	●	●		●		13053	960·610	●	●											HK$6826			●
倚天24P	●	●	●	●		13053	960·700	785	●		●	●	程式轉換							HK$8495			

簡繁體中文卡 ＼ 功能	適用主機PC, XT, AT或COMPATIBLE	24×24字形顯示	無需附加HERCULES GRAPHIC CARD	輸入方法	繁體字庫	簡體字庫	解像度	造字功能	片語功能	ON LINE HELP SCREEN	軟件保護功能	按鍵式純英文及中文狀態轉換	GB231280 內碼 (CCDOS)	NETWORK SUPPORT: ①SPERRY USERNET	②TRANSNET	③ETHERNET	④PCNET	⑤MSNET	⑥NOVELL	定價	附軟件：文書處理	其他軟件：PE2, 123, WS2, d3＋	訓練課程(免費)	按鍵式簡繁轉換
超羣LE-2411	●	●	●	9種	6763	6763	960×700	658	1000 W	●	●	●	●	●	●	●	●	●	●	HK$6260	●	●	●	●
倉頡2025B	●	●		4種		13053	960·610	●	●											HK$6825				
中文之星	●	●				18000		●	●											HK$14040				●

LE-2411九種輸入法爲國標，區位，拼音，部首，電報，首尾，五筆，倉頡，片語

其他型號：LE-8000　16×16 簡、繁體中文卡 $3000.-
LE-5000　16×16 簡體中文卡 $2000.-
LE-2000　16×16 繁體中文卡 $2000.-
LE-1000　16×16 繁體中文碟 $300.-

DISTRIBUTOR：

超羣電子科技公司
中環德輔道中三昌大厦
401-402 室
電話：5-452381,5-441698

SOLE AGENT：

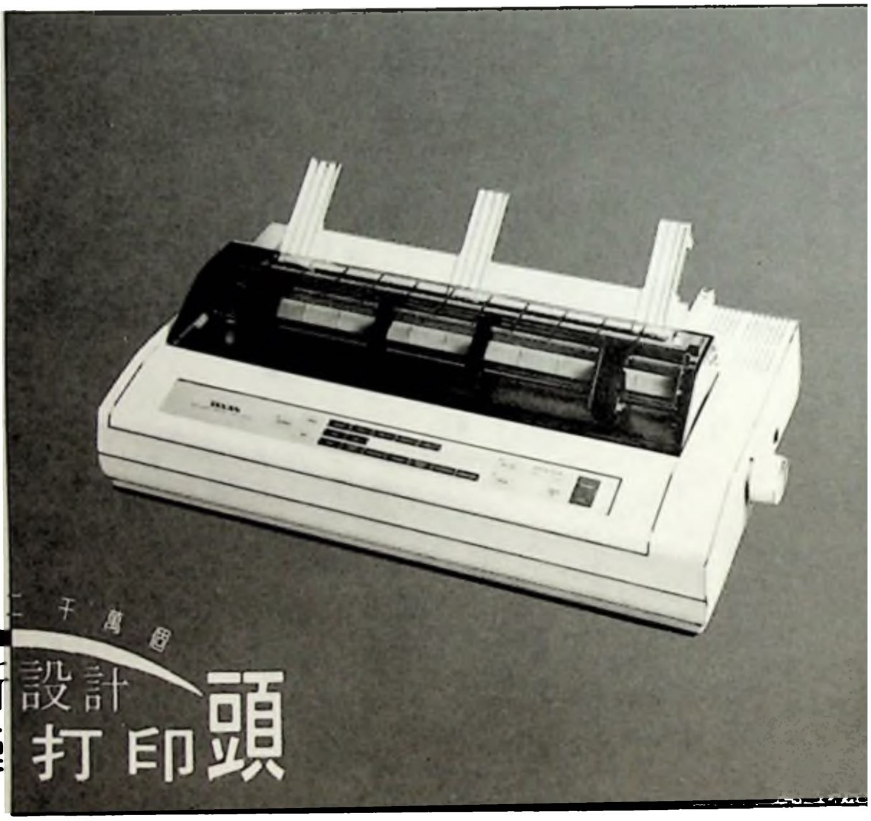
PILOT COMPUTER SYSTEMS
灣仔軒尼詩道302-308號
集成中心2111室
電話：5-730173,5-8921396

TAXAN®

KP-3500

智能超高速打印機

重型，高速，簡易



- 普通字體每秒350字。
- 高質素" NLQ "字體每秒87.5字。
- 簡易板面選擇字體，字體大小及打印方式。
- 軟件完全與IBM - PC[1]及EPSON[2]相容。
- 特長壽重型打印機頭。
- 操作寧靜。

可印字二千萬個

全新設計

特長壽打印頭

1 IBM-PC IS A TRADEMARK OF INTERNATIONAL BUSINESS MACHINES CORP
2 EPSON IS A TRADEMARK OF EPSON CORPORATION

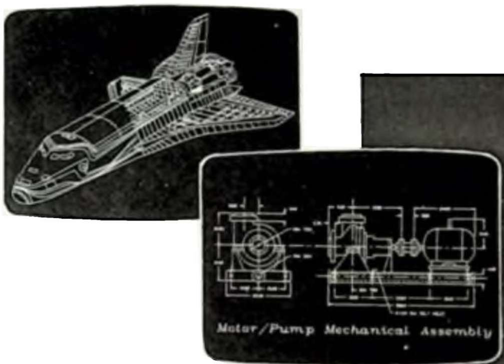
總代理
VINCENT COMPUTER CENTRE LTD.
雲生電腦有限公司
九龍尖沙咀柯士甸路22－26號好兆年行608室
電話：3-7211288-0 電訊：HX 38316 VLCHK 圖文傳真：3-7211828

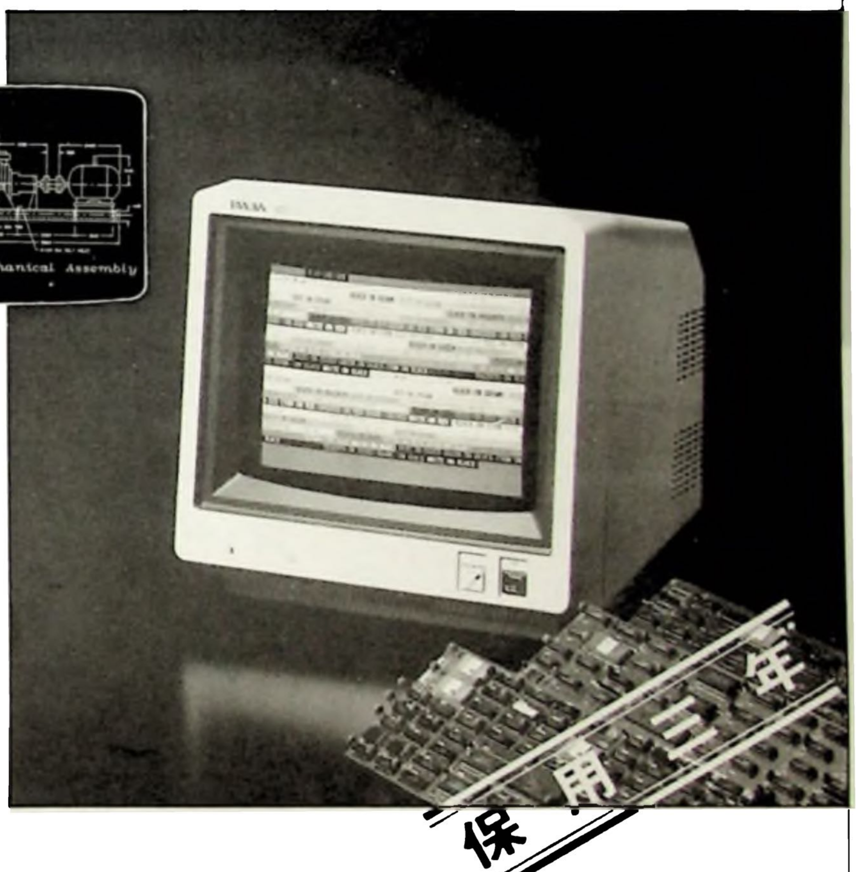
TAXAN®

CAD/CAM 專用彩色電腦螢幕

SUPER VISION IV/IV M

提供特高解像度圖案及文字





- 超高解像度Super Vision IV（720 Dots×400Lines, Dot pitch 0.31mm）、Super Vision IV M（640Dots ×400Lines, Dot pitch 0.37mm）。
- 採用Super High Contrast顯像管，畫面更清晰。
- 顯示高質素文字及圖案，效果可媲美幻燈片。
- 可配用TAXAN KIF-3800、KIF-3810或其他特高解像度彩色圖案界面。
- 適用於IBM®PC 、XT 、AT 或同類型電腦。

顧客注意：

本公司近日發現某些不誠實零售商惡意將TAXAN電腦螢幕之效果調差，藉以推銷其他牌子利潤較優的產品。如顧客遇到上述情況，請往其他商號選購TAXAN產品或親臨本公司參觀操作示範。

總代理：

VINCENT COMPUTER CENTRE LTD.

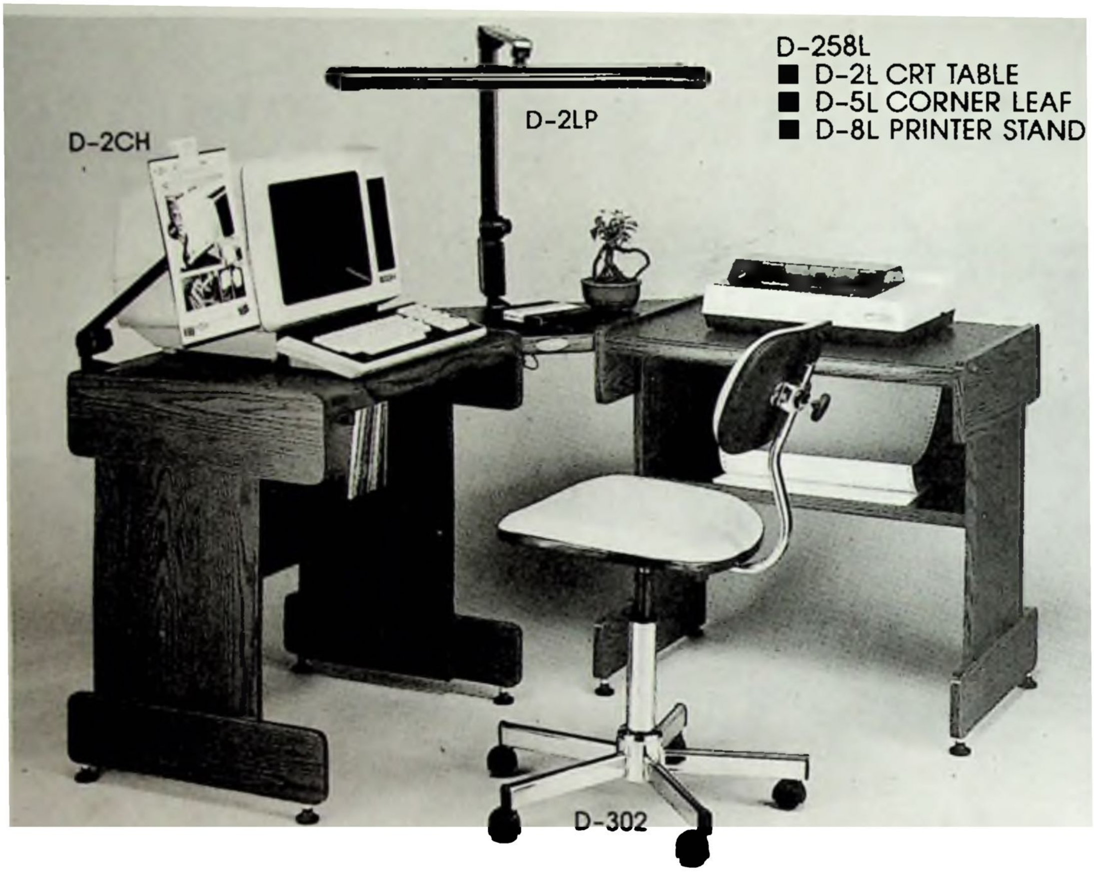
雲生電腦有限公司（請認明有限公司，並無分號）

九龍尖沙咀柯士甸路22-26號好兆年行608室 電話：3-7211288-0 電訊：HX38316 VCC HK

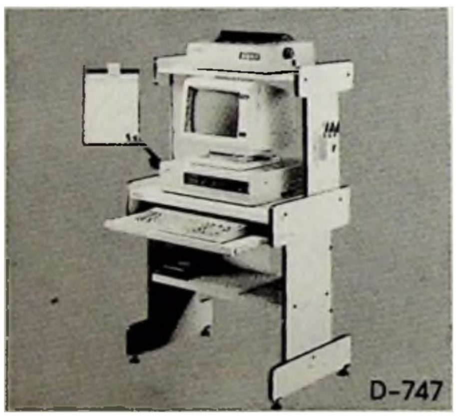
**保用三年乃根據本公司之正式保用條款提供。

K/D Computer Work Stations

- Economical
- Modular
- Do it yourself
- Save Space
- Easy Assembly

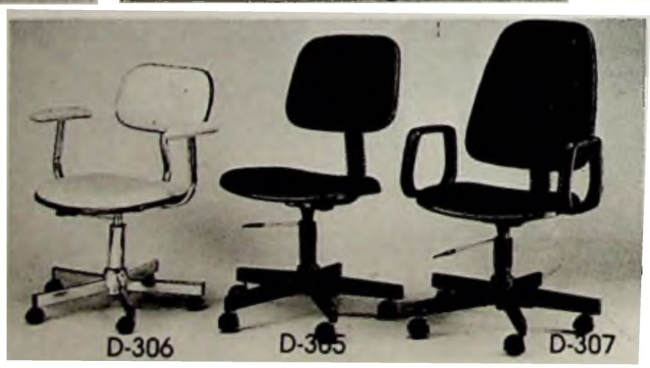












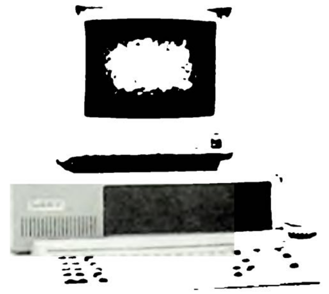


超卓電腦系統有限公司
SUPREMIA SYSTEMS LTD.

Shop No. 20, 54 B/F., Golden Building Commercial Arcade,
146, Fuk Wah Street, Sham Shui Po, Kln., Hong Kong
Tel: 3-601531, 3-612549, 3-869095, 3-617206
Telex: 48258 SUMIA HX Cable: SUPMIASL

電腦功能人皆知
先租後買有保障

每月只須付出三佰餘元，便可使貴公司踏入電腦化坦途。



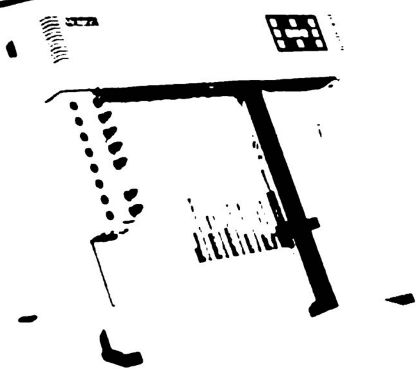
＊歡迎查詢或要求示範＊

本公司推出電腦租機服務，中小企業電腦化好消息。

1. 租用期：三個月，每月由三佰餘元起，若三個月內決定購買，則租金可自買價中減去。
2. 功能：文字處理，會計，生產及營業管理等。
3. 其他服務：可替貴公司訓練電腦人材，確保貴公司能利用電腦解決寫字樓工作。

全港首創開辦

AUTOCAD 2.17 電腦輔助繪圖班



可使您学習到：

(1)繪製平面及立體圖形。
(2)製作動態畫面及幾何圖形。
(3)數位板（DIGITIZER）描圖技巧。
(4)可應用於機械、電子、電機、建築、化學工程……等等。
(5)可配合AUTOCAD軟件於IBM PC/XT/AT 使用。

名額有限，報名從速

本公司附屬機構：**現代電腦訓練中心**

電話：3-601531，3-617206

本中心獲得下列 CAD SYSTEM AUTHORIZED DEALER

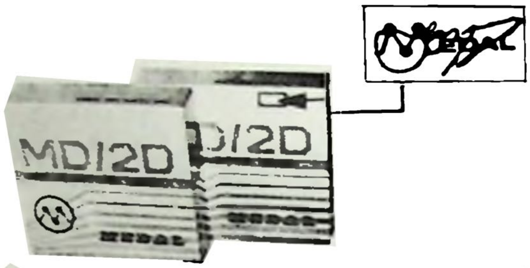
1. PLOTTER　　2. MOUSE
3. DIGITIZER　　4. RGB 高解像 MONITOR

超卓電腦系統有限公司
SUPREMIA SYSTEMS LTD.

九龍深水埗福華街 146 號
黃金商場地庫 20 號室
電話：3-612549，3-601531

美 廸 電 子 公 司

MEDAL ELECTRONICS CO.





MAXELL 2D DISKETTE



MEDAL 2D DISKETTE



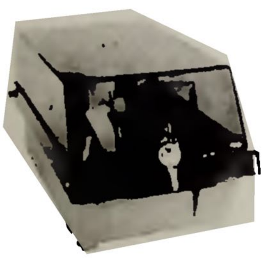
DATALIFE 2D DISKETTE



MEDIA 2D DISKETTE
AUTHORISED DEALER



SUPREME 2D
全港最平



DISKETTE BOX CAN
CONTAINS 100pcs
DH – 100BL



CAN CONTAIN 20pcs
DISKETTF HOLDER



DISKETTE NOTCHER

歡 迎 出 口 批 發 及 零 售

MEDAL ELECTRONICS COMPANY

705, Chong Fat Bldg., 268 Cheung Sha Wan Road, Kowloon.

TEL: 3-864243 (3 LINES) TELEX: 49972 MPBAH HX TELEGRAM: MEDALPROSR

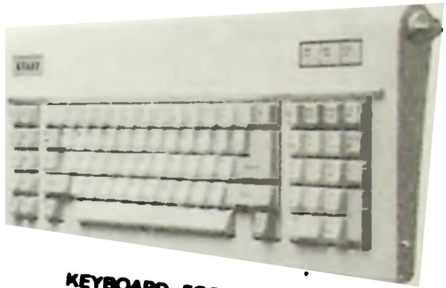
IBM is a registered trademark of International Business Machines

Apple is a registered trademark of Apple Computer Inc.

美迪電子公司
MEDAL ELECTRONICS CO.



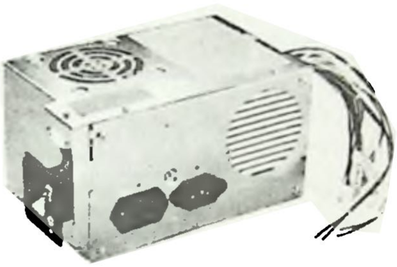
SPEEDY KEYBOARD FOR XT



KEYBOARD FOR AT/XT



5150 KEYBOARD FOR XT





MDL – 220W POWER SUPPLY FOR APPLE COMPUTER



MD – 150W POWER SUPPLY FOR IBM

Features :

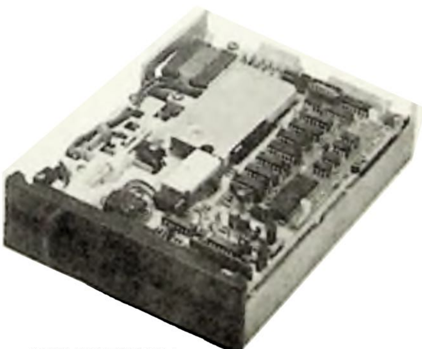
- Overvoltage and overcurrent protection
- EMI conforms to FCC part 15 Subpart J. Class B
- High efficiency and stability
- 110/200V switchable.
- DC colling fan installed
- 8-hour On-Off test, 15 minutes a cycle.
- One year warranty

AT-200W POWER SUPPLY FOR IBM AT

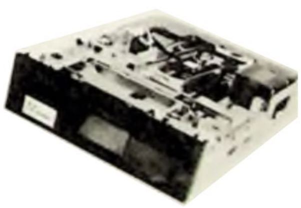
Features.

- HIGH EFFICIENCY
- BUIT-IN EMIFILTER
- LOW OUTPUT RIPPLE
- OVER VOLTAGE PROTECTION
- OVER VOLTAGE PROTECTION
- SHORT CIRCUIT PROTECTION
- DUAL INPUT VOLTAGE 115/230V. AC
- 100% THERMAL CYCLE AND BURN-IN
- MEETS UL/FCC STANDARDS

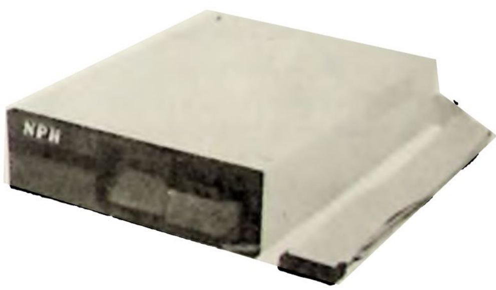
MDL – 220W POWER SUPPLY FOR APPLE COMPUTER



IBM NATIONAL
DRIVE JA – 551 – 45
大量現貨



IBM MEDAL DISK DRIVE



FLOPPY DISK DRIVE FOR APPLE

歡迎出口批發及零售

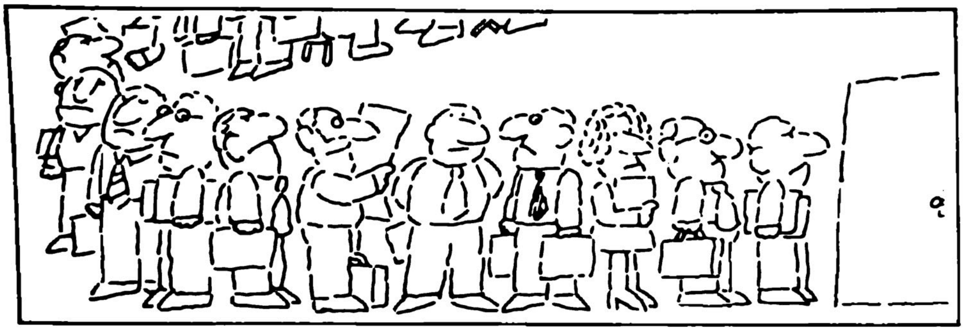
MEDAL ELECTRONICS COMPANY

705, Chong Fat Bldg., 268 Cheung Sha Wan Road, Kowloon.

TEL: 3-864243 (3 LINES) TELEX: 49972 MPBAH HX TELEGRAM: MEDALPROSR-

IBM is a registered trademark of International Business Machines

Apple is a registered trademark of Apple Computer Inc.



市場動態

港大安裝阿波羅「領域」3000工作站

香港大學向阿玻羅電腦有限公司購買四台新一代工作站 ──「領域」(DOMAIN) 3000 系列，用於電腦輔助設計 (CAD) 方面。採用由維多利電腦服務有限公司 (SIA Computer Service (F-ar East) Ltd) 經銷的 CAD 軟件系統。

香港大學所安裝的三台阿玻羅「領域」3000系列工作站是用於八六／八七學年度新開辦的 CA-D 課程。工作站將供建築系及建造系使用，作爲系內設施及輔助教授和研究用。

所安裝的三台 DN 3000 採用 Calcomp 數字化圖形輸入板及多筆繪圖機作爲輸入及輸出設備，並將在 ACROPOLIS/EAGLE 軟件系統上運行。

Eagle 是一通用的二維及三維繪圖系統，附有設計語言系統，使工程師和設計師能設計他們的工程系統。配合 Eagle 系統使用的是 Acropol-is 三維添加模件，用於建造業的建築設計及繪圖、估算及用料單分析。■

改裝收銀機成終端機

一種澳洲製造的單板式微型處理機能將低廉的電子收銀機轉爲精密的銷售點 (POS) 終端機。

由 RSM 資源電腦有限公司研製的零售系統模件 (RSM) 能以電子方法將收銀機日記帳上的大量細目檔案及詳細資料儲存起來；該裝置已連接到 NCR 及 Kingtron 產品系列。

該裝置取代了收銀機內的主邏輯板；除可輸入程序，爲零售商提供所有銷售點功能外，亦具備微型電腦的全面功能。

該公司計劃在全球透過經銷商銷售該裝置，並爲有關用戶提供員工訓練、程序設計及維修服務等。

改裝程序十分方便，經銷商可輕易安裝微型處理機板，然後將改裝好的設備作爲附加產品銷售。

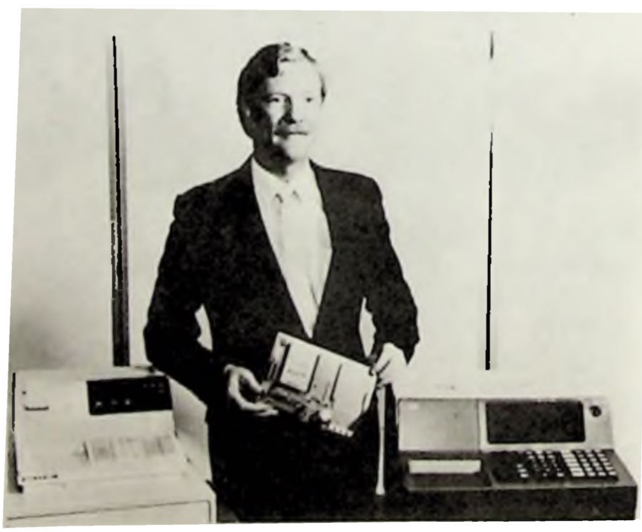
該裝置得到全面的零售軟件程序支援，加強性能，這些軟件由國際程序設計師研製，盡可能符合不同終端用戶及國家的要求。

標準軟件適用於全美國約百分之九十的零售商；由於軟件以參數推動，故此銷售商可以因應買家的規格要求，不單是提供多種選擇。

該裝置具備的功能包括檢查價格、存貨檔案、細目及簡要資料、收費、聯綫發票、可輸入程序的銷售點稅項計算、字母／數字顯示、激光掃描機、打印機、標度及電子轉賬。透過電話調制解調器，可進行遥程診斷。

改裝後的收銀機可以在機間互連通訊或接到主機電腦。由於鍵盤與收銀機的相同，故此改裝後的裝置操作十分方便，最宜用來訓練。該裝置能自動執行更複雜的處理功能，操作人員無需費心。■

圖：澳洲 RSM 資源有限公司研製的單板式微型處理機



新產品介紹

使用80386的AT加速咭

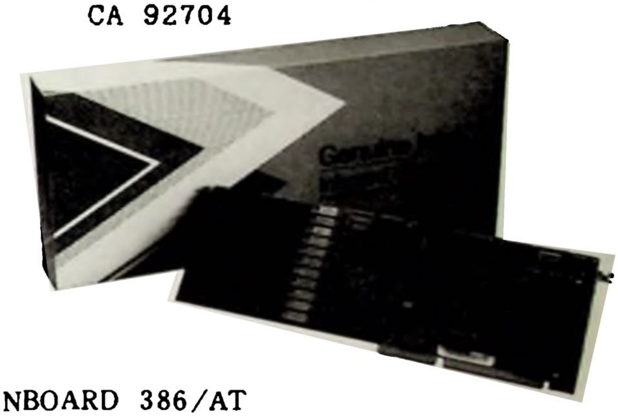
INTEL 所推出的 32 位元微處理器 80386 為整個微電腦界帶來一番新景象，不過現時基於 80386 的微電腦產品仍處於高水平，而且軟件也十分欠缺，暫時最新的發展是使用 80386 的 AT 加速咭，取其較快的工作速度。以下是其中兩家公司的新產品：

INTEL 推出的 INBOARD 386/AT（2495 美元）時鐘速率為 16MHz，能使 AT 的軟件以雙倍速度運行。咭上可加上 1MB 的 RAM（500 美元），用作高速記憶暫存（CACHE）。隨咭附同的是一個控制軟件，把 80386 以虛擬 86 模式工作，容許用家分別執行數個軟件而不必任何改動。

AMERICAN COMPUTER AND PERIPHERAL 所推出的 386 TURBO CARD 售價為 1995 美元，據稱可把 AT 的效率增加四倍。此咭支持特別為 80386 編寫的軟件，以及其他 80286 軟件。咭上附有 1MB 的 RAM，以及特別的倍頻線路，使時鐘速率加倍。■

INBOARD 386/AT $1,995 (U.S.)
$2,495 (with 1 MB RAM)
INTEL CORP. MAIL STOP TOD-07
5200 NE ELAM YOUNG PKWY
HILLSBORO, OR 97124.

AMERICAN COMPUTER AND
PERIPHERAL 386 TURBO
$1,995 (U.S.)
2720 CRODDY WAY, SANTA ANA
CA 92704



圖：INTEL 推出的 INBOARD 386/AT

APPLE或IBM dBASE系統適用

一套完善的中小型工商業管理系統

- 私人開支管理系統

微電腦技術V2•5、V2•6、V2•8

- 庫存管理系統

微電腦技術V2•5、V2•7、V2•8

- 發票及銷售報告系統

微電腦技術V2•5、V2•8、V2•9

歡迎到讀者服務部購買各期磁碟雜誌

高速MODEM新標準CCITT V.32

就目前電腦通訊技術而言，以電話網絡作每秒 9600 位元（BPS）資料傳送是絕對有可能的，美國已有數家生產商專門製造這方面用途的高速調制解調器（MODEM）。但是高速 MODEM 的市場正呈現一片混亂，以致大大影響了在非專業應用的普及。主要的原因是高速 MODEM 要克服下列的困難：

• 現存的高速 MODEM 標準 CCITT（國際電報及電話顧問委員會）V.32 比預期中困難實現在高速 MODEM 上，以致開發成本非常高昂，自然售價亦相應提高了。

• 由於使用V.32 標準的 MODEM 產品較遲才出現在市場（基於上述原因），其間出現了較爲廉價的高速 MODEM，因爲使用不同的發展技術的關係，使到不同高速 MODEM 使用不兼容之標準。

• 電話線路的傳送質素參差不齊，使到高速M-ODEM 不能經常達到應有的傳送速度。

• V.32 標準本身亦存在了漏洞，就算兩部同樣支持 V.32 標準的 MODEM，也會因由不同生產商製造而出現軟件／硬件不兼容的情況。

自然有不少方法被提出來解決上述各項難題，包括了一個全新的不對稱 MODEM 技術規格。至於 MODEM 的未來仍然不明朗，尤其高速 MOD-EM 的方面，沒有人能夠估計到將來的進展及銷量。表一是目前的 CCITT 規格。

不標準的標準

當兩年前每秒 9600 位元傳送速度的 V.32標準被 CCITT 批準之時，不少人都認爲整個 MO-DEM 市場都趨向於能夠支持這標準的高速 MO-DEM 產品。事實上，一家美國主要生產商 CO-NCORD DATA SYSTEMS INC 去年（1986）只售出 1500 個 V.32 MODEM，而另一家生產商 CODEX CORP.亦被逼削減整條生產線。

V.32 標準最特別的地方是不基於較早出現的 CCITT 標準，不過却只出現在實驗室階段，眞正的V.32 MODEM 並未出現。據報不少大公司（包括 AT&T）都在致力發展新一代的 V.32 MOD-EM，但直至現在仍沒有任何產品推出的消息。

造成發展 V.32 MODEM 主要的困難在於所採用“ECHO CANCELLATION”新技術，這項技術的優點是容許在電話線上進行高速雙向通訊，但却大大地提高了成本。就以 CONCORD 公司的V.32 MODEM 產品爲例，3500 美元售價中的四份一便是開發這項技術的成本。話雖如此，V.32 標準還是會繼續下去，因爲對於在電話線上以每秒 9600 位元雙向傳送的需求實在是非常迫切。

售價的考慮

電話網絡的質素也是影響了高速通訊的成功與否，每當電話線路質素下降的時候，V.32 MOD-EM 便需要工作於較低的 4800BPS 速率。如果這種情形經常發生的話，用戶便會懷疑是否値得購買這種高速 MODEM。CONCORD 公司聲稱他們的產品百份之八十時間都能工作於 9600BPS 的速率，但是當需要較低速率之時，便要手動把速率調至 4800BPS。

對於使用 V.27ter 標準（4800BPS）的 M-ODEM 來說，一般的售價爲 1500美元，不過就只能單向傳送，以致效率大打折扣，而且當電話線路未如理想時，傳送速度更降至 2400BPS。

現時工業所接受的 MODEM 售價以每位元／秒爲單位，由 0.25 美元直至 0.3 美元不等。最初的 300BPS MODEM（CCITT V.21，BEL-L 103）直至現時最新的 V.32，每單位售價方面都相去不遠。促成用戶選購更高速 MODEM 的動機是爲了節省電話費方面的開支，尤其經常作越洋通訊的話高速（9600BPS 或更高）MODEM的優點就更加明顯了。

高速MODEM帶來的問題

剛才提到當電話線路質素下降的時候，傳送速度便需要降低，這時因爲通訊過程會出現很多中斷，看軟件使用的通訊協定而定。較爲複雜的是延遲

與 TIME-OUT 的處理，使到原來的 9600 BPS 傳送速度降至 7500BPS 左右。似乎現時的通訊軟件及協定應該認真地改良，以適應高速通訊方面的需要。

幸好，已有跡象 V.32 MODEM 的售價會下降，主要原因是目前正處於發展階段而非正式生產階段。只要能掌握到製造的技術，以集成電路來實現並不是件困難的事。不過部份公司却需要收回發展成本，故除非是外國公司直接競爭，高速 MOD-EM 的售價短期內都不會有大變動。

另一項好消息是 AT&T 公司正逐漸改善全美國的電話網絡系統，包括更新在 60 年代安裝的回聲取消器（ECHO CANCELLERS），這個計劃完成後高速通訊的傳送效率便會大大地提高。不過現時發展的 V.33 MODEM 是附有回聲消除的設計，到時便沒有回聲可消除，使 MODEM 不斷地產生回聲消除之訊號而失效。AT&T 正準備使網絡接受 MODEM 傳過來的特別訊號（一個 2100Hz 音調，每 20 微秒作 180 度相位轉變）來關閉回聲消除器，以上的步驟在 CCITT V.25 出現時已被提出。我們相信電話通訊網絡的改善是世界性的趨勢，因為通訊系統是社會進步的原動力。

雖然 V.32 已被公認為高速通訊的標準，但仍然面對其他生產商的競爭，V.32 以外的 MODE-M 產品通常自成一個標準，它們之間亦不能兼容。造成這等產品吸引的原因是它們的低售價，與 CONCORD 或 CODEX 公司的標準 V.32 MO-DEM 相比足足平了一半。最功功的非 V.32 M-ODEM 生產商為 TELEBIT 公司，該公司其中一個產品 TRAILBLAZER MODEM 據稱可以造出每秒 18000 位元的傳送速度。表三是目前各廠商的產品規格比較。

電腦之間高速通訊的需求是非常殷切的，正如家用錄影機（VCR）一樣，就算你明知價格會一直下降，但現在你確有需要立即使用便會購買。■

表一：目前 CCITT 各個標準一覽表

CCITT (BELL) SPECIFICATION	BIT RATE (BIT/S)	FULL DUPLEX OVER 2-WIRE DIAL-UP?	FALLBACK DATA RATE (BIT/S)	SYNCHRONOUS/ASYNCHRONOUS: DTE TO MODEM	SYNCHRONOUS/ASYNCHRONOUS: BETWEEN MODEMS
V.21(BELL 103)	300	YES	0-300	ASYNC	ASYNC
V.22(BELL 212A)	1,200	YES	600	ASYNC/SYNC	SYNC
V.22BIS	2,400	YES	1,200	AYSNC/SYNC	SYNC
V.26BIS	2,400	NO	1,200	SYNC	SYNC
V.26TER	2,400	YES	1,200	SYNC	SYNC
V.27TER	4,800	NO	2,400	SYNC	SYNC
V.32	9,600	YES	4,800	SYNC	SYNC
ASYMMETRICAL (PROPOSED)	9,600/300	YES	-	ASYNC/SYNC	SYNC

表二：高速 MODEM 生產商之地址及其產品

VENDOR	MODEM	DIAL-RATE	FULL DUPLEX	PRICE	ADDRESS
CONCORD DATA SYSTEMS	CDS V.32	9.6 KB/S	YES	$3,495	397 WILLIAM STREET, MARLBOROUGH, MASSACHUSETTS 01752
CORDEX CORP. DEPT.	2260	9.6 KB/S	YES	$3,495	707-84, CABOT BOULEVARD MANSFIELD, MA 02048
ARK ELEC. PRODUCTS INC	ARK 12000	12 KB/S	NO	$2,900	1500 WEST NASA BOULEVARD PO BOX 2169, MELBOURNE, FLORIDA 32902-2169

通訊軟件加上ANSI色彩介面

我們曾經在過去的數期裏介紹過終端機程式的製作過程及它的技巧，由於時下 COLOR GRAPHIC 咭是非常流行，而很多用戶都使用彩色顯示器，我們今期便介紹一個可以令終端機送出 ANSI 色彩標準的組合語言介面，這樣可令通訊更加色彩繽紛，多姿多采。

雖然這個介面所設計的標準是配合 BASIC 編譯器的，但仍然可以使用在 PASCAL 和 C 之上。在 BASIC 上呼叫時只要採用以下的格式指令可以：

CALL ANSI (A$，C.C，C.L)

其中 A$ 是要輸出的字串，C.C 是列數和 C.L 是行數。

在整體配合上 ANSI.ASM 可以先用 MACRO ASSEMBER 4.0 進行編譯：

MASM ANSI

之後會得出一個 ANSI.OBJ 檔，然後你可以將終端機程式修改，並進行編譯。

BASCOM TERM.BAS /X/N/O/C：4096。這樣便得到一個 TERM.OBJ 檔案。此時便可以將以上兩個檔案連結得到最後的可執行檔。

LINK TERM＋IBM.COM＋ANSI，TERM.EXE，BASCOM20 便可以得到一個 TERM.EXE 檔，可以處理 ANSI 標準的色彩了。

即期微電腦技術程式磁碟（IBM 版本）收錄了本文的原程式檔 ANSI.ASM，以及可行程式檔 TERM.EXE。■

列表一　ANSI.ASM

```

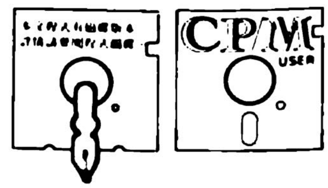
ANSI_PRNT SEGMENT PUBLIC 'CODESG'              ;By  COMPUHOUSE
          ASSUME CS:ANSI_PRNT                  ;
          PUBLIC ANSI                          ;

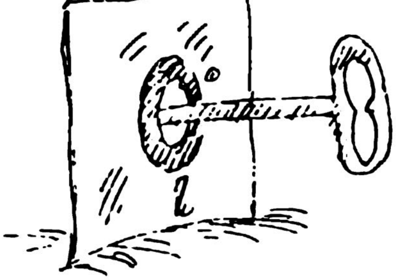
;
;
;
;
;
STRG_LEN          DW 0                         ;CHANGED TO LENGTH OF STRING PASSED
VID_PAGE          DB 0                         ;Active video page
;
;
ANSI      PROC    FAR
          PUSH    BP
          MOV     BP,SP
;
          MOV     SI,10[BP]            ;GET STRING DESCRIPTOR
          MOV     BL,[SI+ 2]           ;REARRANGE LOW/HIGH BYTES
          MOV     BH,[SI+ 3]           ;NOW BX HOLDS THE ADDRESS OF THE STRING
          MOV     AX,[SI]              ;GET STRING LENGTH
          ADD     AX,BX                ;ADD INITIAL OFFSET (BX) TO LENGTH
          MOV     STRG_LEN,AX          ;STORE OFFSET PLUS LENGTH

          PUSH    BX
          MOV     AH,15                ;Get current video state
          INT     10H
          MOV     VID_PAGE,BH          ;Save it
          POP     BX

          MOV     AH,02                ;SET UP FOR FUNCTION CALL 02H
LOOP:
          MOV     DL,[BX]              ;SET DL TO CHARACTER TO PRINT
          PUSH    DX                   ;Save the character in AX 'till we check..
          CALL    WHERE_ARE_WE         ; where the cursor is.......
          CMP     DH,17H               ;Row 23?
          JL      NOPE                 ; Jump if less......
          CMP     DH,18H               ;Row 25?
          JNZ     NEXT2                ;Keep checking if not
```

```
            DEC     BX              ; Else backup one character
            JMP     SCROLL2         ; And go scroll the screen
;
NEXT2:      POP     DX              ;And restore the stack to where it was
            CMP     DL,0AH          ;Do we have a line feed?
            JZ      SCROLL          ; Yup - scroll this sucker!
            CMP     DL,0DH          ;  How about a carriage return?
            JNZ     NOPE1           ;  Nope - just go display it.......
            INC     BX              ;  Yup - see if next char is a line feed
            MOV     DX,[BX]
            CMP     DL,0AH          ;  Well, is it?
            JZ      SCROLL          ;  It sure is - let's go scroll
            DEC     BX              ;  Oops - just a carriage return
            JMP     SCROLL          ;  But let's go scroll it anyway
NOPE:       POP     DX
NOPE1:      INT     21H             ;Else just display it - same old boring stu**
SKIPIT:     INC     BX
            CMP     BX,STRG_LEN
            JL      LOOP            ;LOOP UNTIL ALL CHARS PROCESSED
;
            MOV     AH,03           ;SET UP FOR ROM-BIOS CALL (03H)
            MOV     BH,VID_PAGE     ;TO READ THE CURRENT CURSOR POSITION
            INT     10H             ;  DH = ROW   DL = COLUMN
            INC     DH              ;ADD 1 TO ROW (BECAUSE TOP OF SCREEN = 0)
            INC     DL              ;ADD 1 TO COL (BECAUSE POS 1 = 0)
;
            MOV     SI,[BP]+ 8
            MOV     [SI],DH         ;PASS BACK ROW COORDINATE
            MOV     SI,[BP]+ 6
            MOV     [SI],DL         ;PASS BACK COLUMN COORDINATE
;
            POP     BP
            RET     6
ANSI        ENDP
Where_Are_We:                     ;Get the current cursor position
            PUSH    AX              ;Save the registers
            PUSH    BX
            PUSH    CX
            MOV     AH,03           ;SET UP FOR ROM-BIOS CALL (03H)
            MOV     BH,VID_PAGE     ;TO READ THE CURRENT CURSOR POSITION
            INT     10H             ;  DH = ROW   DL = COLUMN
            POP     CX              ;Restore the registers
            POP     BX
            POP     AX
            RET                      ;And go back from wence we came
SCROLL2:    POP     DX              ;Put the stack like it was
SCROLL:     CALL    SCROLLIT        ;Scroll the screen
            JMP     SKIPIT
SCROLLIT:   PUSH    AX              ;Save the registers that will be affected
            PUSH    BX
            PUSH    CX
            PUSH    DX
            PUSH    BP
            MOV     AH,2            ;Now set cursor position to 23,0
            MOV     DX,1700H        ;so we can get the proper character
            MOV     BH,VID_PAGE     ;attribute
            INT     10H
            MOV     AH,8            ;Get the current character attribute
            MOV     BH,VID_PAGE
            INT     10H
            MOV     BH,AH           ;Transfer the attribute to BH for next call
            MOV     AH,6            ;Otherwise scroll 23 lines
            MOV     AL,1            ; Only blank line 23
            MOV     CX,0000H        ; Begin scroll at position 0,0
            MOV     DX,174FH        ; End scroll at Line 23, Col 79
            INT     10H             ; And do it.......
            MOV     AH,2            ;Now set cursor position to 23,0
            MOV     DX,1700H
            MOV     BH,VID_PAGE
            INT     10H
            POP     BP
            POP     DX              ;Restore the stack like it was
            POP     CX
            POP     BX
            POP     AX
            RET
ANSI_PRNT   ENDS
            END
```





CP/M 檔案加上密碼保護

- 黃偉江 -

大家都知道 CP/M-80 系統(例如 APPLE CP/M)是一個開放式架構的操作系統，所有檔案都沒有保安的設計。萬一有一天你不慎把自己精心設計的程式流傳到其他人手上，到時你的損失便十分巨大了。以下筆者所介紹的程式 PASSWORD.COM 能夠將所有 CP/M 可執行檔(.COM) 加上密碼保護，執行該檔時需輸入正確的密碼才能繼續執行。

使用本程式之時要特別小心，因為當一個檔案加上密碼後便不能解拆(至少暫時不能)。假如你忘記了密碼的話便十分不幸了，本程式使用了十分特別之編碼技巧，並不如其他密碼系統般有漏洞，就算你使用 DUMP 指令也看不出密碼。

程式的使用方法很簡單，假如你有購買即期程式磁碟(CP/M 版本)的話，BOOT 起 CP/M 系統後，鍵入：

A>PASSWORD 可執行檔案名

Password ?

跟着便可輸入密碼了，切記密碼不能長於 6 個字符，而且更要注意大小楷的使用。再次執行該檔案時須正確地輸入密碼，否則只會回到 CP/M 系統。

很明顯，PASSWORD 是一個十分實用的程式，不過為安全起見，使用 PASSWORD 之前該把你的檔案作 BACK-UP，以及避免將 PASSWORD 這程式落在他人之手。■

列表一：PASSWORD.ASM 原程式

```
;         title   'PASSWORD.ASM'
;
;                 PASSWORD.ASM
;                 Version 3.0
.          By Compu-Link Magazine  (c) 1987
;
;
;EQUATES
rdchar:  equ      1
mesout:  equ      9                   ;BDOS functions
incon:   equ      10
open:    equ      15
close:   equ      16
delete:  equ      19
read:    equ      20
write:   equ      21
setdma:  equ      26
;
cr       equ      0dh                 ;ascii values
lf       equ      0ah
eos      equ      '$'
;
boot     equ      0;4200H             ;0 for standard CP/M
                                      ;4200H for ALT. CP/M;
bdos     equ      boot+5
fcb      equ      boot+5ch
defbuf   equ      boot+80h
tpa      equ      boot+100h
stack    equ      tpa                 ;out of way of tpa
;
         org      tpa

;
;
start:   lxi      h,0                 ;save stack pointer
         dad      sp                  ;put stack in hl
         shld     oldstack-offset     ;save it
         lxi      sp,stack            ;get new stack
;
; stack saved so program can return to CCP without
; intervening warm start.
;
         lda      fcb+9               ;get first char of extension
         cpi      ' '                 ;if ' ' then change to .COM
         jz       notype
         cpi      'C'                 ;If there is an extension.
         jnz      notright            ;make sure it's .COM
         lda      fcb+10              ;check second letter
         cpi      'O'
         jnz      notright
         lda      fcb+11
         cpi      'M'                 ;last letter
         jz       iscom               ;if it is a COM, then cont.
notright
         call     endmes              ;it's not a com file, so tell


         db       cr,lf,'Must be a command (.COM) file'
         db       cr,lf,eos
;
endmes
         pop      d                   ;get address of message
         mvi      c,mesout            ;PRINT STRING command
         call     bdos                ;print error message
;
finish:  lhld     oldstack-offset     ;get old stack
         sphl                         ;put it in HL
         ret                          ;return to CP/M
;
notype:  mvi      a,'C'               ;if there was space, change
         sta      fcb+9               ;to COM
         mvi      a,'O'
         sta      fcb+10
         mvi      a,'M'
         sta      fcb+11
;
iscom:   lxi      d,buffer-offset     ;point to where program goes
         mvi      c,setdma            ;SET DMA command
         push     d                   ;save it
         call     bdos                ;and tell CP/M
         mvi      a,0                 ;zero record count
         sta      fcb+32
         mvi      c,open              ;OPEN file command
         lxi      d,fcb               ;load address of FCB in DE
         call     bdos                ;Open file
         inr      a                   ;successful?
         jnz      openok              ;if so, then continue
         call     endmes              ;if not, then tell
;
         db       cr,lf,'Cannot open file',cr,lf,eos
;
openok:  pop      d                   ;get starting dma back
rloop:   mvi      c,setdma            ;and set it in loop
         push     d                   ;save it
         call     bdos
         lxi      d,fcb               ;point to FCB
         mvi      c,read              ;READ sector command
         call     bdos                ;do it
         pop      d                   ;get DMA address back
         ana      a                   ;EOF?
         jnz      doneread            ;if so, then ask for password
         lxi      h,80h               ;length of sector
         dad      d                   ;bump DMA
         xchg                         ,put new address in DE
```

```
        jmp     [illegible]             ;and read some more
;
doneread
        xchg                            ,dma ==> hl
        shld    endprog-offset          ,save last address
gpasag  call    getpas                  ;print password message


[illegible] db  Password? ',eos

getpas  pop     d                       ;get address of message
        mvi     c,mesout                ;PRINT STRING function
        call    bdos                    ;print it
        lxi     d,defbuf                ;point to default buffer
        mvi     a,[illegible]           ;tell CP/M max chars
        stax    d                       ,put it there
        mvi     c,[illegible]           ;READ LINE command
        call    bdos                    ,do it
        lxi     h,defbuf+1              ,point to length
        lxi     d,password-offset       ,point to storage
        mov     a,m                     ,get length
        ana     a                       ;set flags
        jz      gpasag                  ,if 0 then ask again
        inr     a                       ;plus 1 for length byte
        mov     b,a                     ;put length in B
        xra     b                       ,cancel first xra in loop
        mov     m,a                     ;and put it back
mploop  mov     a,m                     ;get char
        xra     b                       ;encrypt it
        stax    d                       ;save it
        inx     h                       ;increment pointer
        inx     d                       .
        dcr     b                       ;decrement length
        jnz     mploop                  ;if not zero, then next char
        xra     a                       ;zero a
        sta     fcb+12                  ,zero bytes in FCB
        sta     fcb+14
        sta     fcb+32
        mvi     c,open                  ,OPEN file command
        lxi     d,fcb                   ;point to FCB
        call    bdos                    ,open the file
        lxi     d,nstart                ,point to new program start

        push    d
wloopl  pop     d                       ;get DMA
        push    d                       ,put it back on stack
        mvi     c,setdma                ;SET DMA command
        call    bdos                    ;tell CP/M
        lxi     d,fcb                   ;point to FCB
        mvi     c,write                 ,WRITE SECTOR command
        call    bdos                    ;do it
        pop     h                       ;get DMA address from stack
        lxi     d,80h                   ;length of sector
        dad     d                       ;HL has new DMA
        push    h                       ,put it on stack
        mov     a,h                     ;this is to get 2's complement
        cma                             ;of address  We are subtracting
        mov     d,a                     ;the current address from the
        mov     a,l                     ;high address. If the high byte
        cma                             ;<1 , we are done
        mov     e,a                     .
        inx     d                       ,Now 2's comp. of address in DE
        lhld    endprog-offset          ;get ending address

dad     d                       .Subtract (add 2's comp)
        mov     a,h                     ,get high byte
        inr     a                       ,is it FF (-1)?
        ana     a                       ;set flags
        jnz     wloopl                  ,if not, write another sector

        mvi     c,close                  That's it. Close the file
        lxi     d,fcb                   .point to FCB
        call    bdos                    .do it
        jmp     finish                  goto finish


nstart
offset  equ     tpa-nstart
.
:       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
.       %% WARNING -                                      %%
:       %% From now on, all labels are in                 %%
;       %% the form:                                      %%
.       %%      LABEL   EQU  $+OFFSET                     %%
.       %%  This is to allow the program to run at100H    %%
.       %% when it is saved by the earlier portion.       %%
.       %%  ALL new labels added MUST be in the form      %%
;       %% LABEL   EQU  $+OFFSET for this program to work %%
:       %% properly                                       %%
:       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
;This is portion of the program is placed at the beginning
;of the program to be PASSWORDed  When it is executed, it will
;ask for a password  If the password is incorrect, the program
,warm starts  If the password is correct, the program is moved
;to the TPA and executed.
.
        lxi     h,0                     ;save stack pointer
        dad     sp                      :stack is in HL
        shld    oldstack                ;save it
        lxi     sp,stack                ,get new stack
        call    otpw                    :print password message

        db      cr,lf,'Password?
        db      eos

otpw    equ     $+offset
        pop     d                       ,get address of message
        mvi     c,mesout                :PRINT STRING command
        call    bdos                    .print it

        lxi     d,newbuf+1              .point to storage area
        xra     a
        mov     [illegible]             ;init counter
        stax    d                       .# of chars typed (for compare)
        inx     d                       .and point to string

   we don't want to allow 'lookers' to see the password, so call
   bios instead of bdos to prevent echo   also allows control chars


        lhld    boot+1                  ;get addr of warm start routine
        push    d
        lxi     d,6                     ;+ offset to kbd input routine
        dad     d
        pop     d
        shld    bconin+1                ;now init'd to bios conin

otpwl   equ     $+offset
        push    b                       ;save counter..
        push    d                       :..& pointer
bconin  equ     $+offset
        call    $-$                     ;to be filled in by above routine
        pop     d                       ;restore pointer
        pop     b                       :..& counter
        stax    d                       ;store char
        inx     d                       :bump pointer..
        inr     b                       . .& counter
        cpi     cr
        jnz     otpwl                   :loop until cr typed
: B has the length of password furnished + 1 for count.
        lxi     h,password              ;point to actual password
        lxi     d,newbuf+1              ;point to user's input
        xra     a                       ;this 0 xra with length in B
        stax    d                       ,should equal count in passwd
:
: first char compared is a count of # of chars in password,
: followed by password itself
;
clp     equ     $+offset
        ldax    d                       ;get char
        xra     b                       ;encrypt it
        cmp     m                       ;are they the same?
        jnz     boot                    ;if not, restart
        inx     h                       ;point to next characters
        inx     d                       .
        dcr     b                       ;decrement length
        jnz     clp                     ,if not done, then loop
:
: Now we move a segment of code to a part of the default
, buffer. This segment moves the actual program down to the
: TPA
;
        lxi     h,nmv                   ;point to code
        lxi     d,defbuf+20h            ;point to new postion
        mvi     b,nmlen ;length
;
move    equ     $+offset
        mov     a,m                     ;get byte
        stax    d                       .save it
        inx     d                       ;point to next addresses
        inx     h                       .
        dcr     b                       .decrement length
        jnz     move                    :if not done, loop
        jmp     defbuf+20h              .go to segment

nmv     equ     $+offset                :segment that gets moved
        lhld    oldstack                .get stack pointer
        push    h                       :save it on stack
        lxi     h,buffer                .get start of actual program
        mov     a,h                     ,We have to compute the length
        cma                             :and because X-Y equals
        mov     d,a                     :X + Two's complent(Y), we have
        mov     a,l                     .to find the 2's comp  of the
        cma                             :first address
        mov     e,a                     .
        inx     d                       ;Y is in DE
        lhld    endprog .get last address
        dad     d                       ;subtract (add 2's comp)
        mov     b,h                     ;put length in BC
        mov     c,l                     ;
        lxi     d,tpa                   :point to TPA
        lxi     h,buffer                :point to first address
nmlp    equ     defbuf+20h+$+offset-nmv
        mov     a,m                     ;get byte
        stax    d                       .save byte
        inx     h                       :increment address
        inx     d
        dcx     b                       .decrement length
        mov     a,b                     .check for zero left
        ora     c                       :Are we done?
        jnz     nmlp                    .if not, loop some more
        pop     h                       ;get stack from stack
        sphl                            .put stack in SP
        jmp     tpa                     .run program

nmlen   equ     $+offset-nmv    ;length of segment

.
password        equ     $+offset :password storage
        db      0

.12/10    changed to allow use of tbuf as input area
.
newbuf  equ     $+offset        .Users input buffer
        db      10H,0,'
.
oldstack        equ     $+offset :place for stack
        ds      2
:
endprog equ     $+offset .place for address
        ds      2
.
buffer  equ     $+offset        .where actual program goes
        end
$
```

爲APPLE版本TURBO PASCAL

增加音樂及說話功能

陳彥

今次筆者爲大家介紹一個在 TURBO PASCAL 下使用音樂及講說話的程式。其中有 SPEAKER.PRO，MUSIC.PRO 分別是說話和奏音樂的主要程序，SPEAKER.PAS 是一個模擬 APPLE 軟件「爲你喉舌」的程式，MUSIC.PAS 是作曲的程式，而 TALK & MUS.PAS 是一 DEMO 程式，示範將 SPEAKER.PAS 和 MUSIC.PAS 連結在一起執行。另有四個 FILES：WELCOME.TXT 是儲存着 DEMO 程式開始時所說的歡迎字句，BACH.DAT 是 DEMO 中所奏的音樂程式，MUSIC.SHP 及 MUSIC.DAT 是 MUSIC.PAS 的資料檔案。

TALK&MUS.PAS 程式執行時，會首先奏出一段音樂，然後用廣東話說出歡迎的字句，繼而出現主目錄，可按鍵選擇 TURBO SPEAKER 或 MUSIC EDISOR。以下是它們的使用方法。

TURBO SPEAKER

程式開始會首先出現一個 MENU，其中包括：

1P —— 讀出一個以音標組成的字串。
1S —— 讀出一個字串。
1F —— 讀出一個 TEXT FILE。
1C —— 改變說話的速度或音調的高低。
1Q —— 離開 TURBO SPEAKER。

MUSIC EDITOR

程式開始時亦會出現一個 MENU，如下：

(1)清洗記憶體中的資料。
(2)將輸入的檔案讀進記憶體。
(3)在高解像畫面上編輯一首樂曲，功能鍵如下：

"I"—— 把游標上升一格，亦即是上升一個調子。

"M"—— 把游標下降一格，亦即是下降一個調子。

"J"↑H (←)—— 游標後退一列，並清洗列內資料，但若在最左的一列，便不能再後退了。

"1"—— 螢光幕會在游標的位置出現一個全音符。"2"出現一個二分音符，"3"出現一個四分音符，餘此類推。一直至到"5"（十六分音符）。

"!"(SHIFT1)—— 游標所在位置出現一個全音符及一符點。即相等於一個全音符加一個二分音符（符點代表所屬音符的一半）。"""(SHIFT2）則出現一個二分音符加一四分音符，餘此類推。直至"$"(SHIFT4)。

"S" 把游標前一格的音符加一個 SHARP，即升半調。

"F" 把游標前一格的音符加一個 FLAT，即降半調。

"P" 把目前記憶體中的音符奏出來。

"Q" 自 EDITOR 中回到 MUSIC EDITOR MENU。

(4)把記憶體中的資料 SAVE 到磁碟裏。
(5)奏出記憶體中的音符。
(6)離開 MUSIC EDITOR。

SPEAKER·PRO的內容

■PROCEDURE UPPERCASE (STRG : STRING255);
將字串 STRG 內所有字元轉爲大寫。

■PROCEDURE NORMAL;
相等 BASIC 指令中的 NORMAL。

■PROCEDURE INVERSE;
相等 BASIC 的 INVERSE。

■PROCEDURE SAY (SAY_ST : STRING255);
把 SAY_ST 字串說出來。例：SAY ("HEL

-LO”)。

■ PROCEDURE PRONOUNCE (PRONOUNCE ST : STRING255);
把 PRONOUNCE_ST 字串說出來，PRONOUNCE_ST 是由音標組成的字串。例：PRONOUNCE (“KUMPYUW4TER”);

■ PROCEDURE SAY_FILE (FILENAME : FILENAME : FILESTRING;
MODE : BYTE);
把以 FILENAME 爲名稱之 TEXT_FILE 讀讀來。MODE 參數必須是 1 或 2，1 表示把 FILE 以普通方法讀出，2 表示把 FILE 以拼音方法說出。例：SAY_FILE (“WELCOME.TXT”);

■ PROCEDURE READ_TALK_ROUTINE;
將程式所需的資料檔案(TALK1.DAT 或 TALK2.DAT)讀進記憶體中，TALK1.DAT 及 TALK2.DAT 即是在「爲你喉舌」磁碟中拆出來的 TALK1 及 TALK2 (第 26 期的電腦時代有詳盡的介紹)，讀者可利用 CP/M 系統磁碟中的 APDOS 把 DOS 中的 FILE 轉到 CP/M 碟中。使用此程序要留意 BLOCKREAD 中的參數，如果要讀取 TALK1.DAT，參數值應是 76，若是 TALK2.DAT，參數值應是 124。如果只利用拼音的功能，那便只須將 TALK1.DAT 讀入記憶體，若要同時有拼音及說英文單字的功能便要利用到 TALK2.DAT 了。

MUSIC · PRO的內容

變數陣列：

LEN_ARY 及 PIT_ARY 是用來儲存樂曲的資料，在 LOAD，SAVE 程序中都要使用。

■ PROCEDURE POKE_SUB (ADDRESS : INTEGER;
CODE : STRING255);
將 CODE 中的資料放進記憶體 ADDRESS 的位置中，CODE 中每一個 BYTE 都要“1” 分隔開，最後一個 BYTE 後都要加上“1”，例如：POKE_SUB ($1FD0，“$11/$FF/$5E/”);即是把 $11 POKE 入 $1FD0，$FF POKE 入 $1FD1，$5E POKE 入 $1FD2 記憶體中。

■ PROCEDURE POKE_SOUND_ROUTINE;
把奏音樂的機械語言 POKE 進記憶。在利用 PROCEDURE SOUND 前必須先執行本程序。

■ PROCEDURE SOUND (LEN，PIT : BYTE);
利用參數 LEN，PIT 發出聲音，LEN 是聲音的長度，PIT 是聲音的高低，兩個參數值都是在 0 至 255 之間。例：SOUND (4，127);

■ PROCEDURE LOAD_FILE (NAME : FILESTRING;
VAR COUNTER : INTEGER);
把以 NAME 爲名的音樂檔案讀進 LEN_ARY 及 PIT_ARY 中。COUNTER 參數是用作記錄資料被讀進的個數。使用時除 NAME 以外，還須附上壹個 INTEGER 型別的變數。
例：LOAD-FILE (“BACH.DAT”，I); I 輸入時爲一任意 INTEGER 變數，回返值爲讀進資料的個數。

■ PROCEDURE SAVE_FILE (NAME : FILESTRING;
VAR FILE_LEN : INTEGER);
在 LEN_ARY 及 PIT_ARY 陣列變數中，把 FILE_LEN 個資料存入以參數 NAME 爲名的檔案。LEN_ARY 及 PIT_ARY 陣列變數分別是樂曲中每一音符的相對長度及高低，例：SOUND (LEN_ARY [1]，PIT_ARY [1]);即是演奏樂曲中第1個音符，如此類推。FILE_LEN 是 SAVE 音符的個數(一音符包括其長度及高低兩個元素)。
例：SAVE_FILE (“MUS.DAT”，2);是把 LEN_ARY 及 PIT_ARY 頭兩個數值(共 4 BYTE) SAVE 到 MUS.DAT 檔案中。

■ PROCEDURE PLAY_FILE (NAME : FILESTRING);
奏出參數 NAME 所指的檔案，在演奏前本程序會自動呼叫副程序 LOAD_FILE。例：PLAY FILE (“BACH.DAT”);

COMPILE程式

由於 TALK1.DAT 的機械語言是由 $3000 (6502 的位址 $4000) 開始，故此與高解像頁有衝突，請注意在每次利用高解像之後都應把 TALK1.DAT 或 TALK2.DAT (TALK2.DAT 也是由 $3000 開始的)讀進記憶體，即是重新呼叫副程序 READ_TALK_ROUTINE。TALK1.DAT 的長度爲 $2600，因此如要利用它，在 COMPILE 時便要把 START ADDRESS 定爲 $5600；另外，TALK2.DAT 的長度爲 $3E00，所以在 COMPILE 時 START ADDRESS 要設爲 $6E00。

爲APPLE版本TURBO PASCAL增加音樂及說話功能

至目前爲止，筆者已爲各位介紹了在 CP/M 80 下的 TURBO PASCAL 利用高解像和講說話及音樂的擴充，如日後另有發現，再與各位研究。再會！

本文之程式檔案頗多，而且部份只屬資料檔，所以這裡只刊出較爲重要的 TALK&MUS.PAS(列表一)、SPEAKER.PAS(列表二)以及 MUSIC.PAS(列表三)。大家可從即期微電腦技術程式磁碟(CP/M 版本)找到其餘的檔案。其中 MUSIC EDITOR 程式需要使用 APPLE 的高解像畫面，只適用於雙插頭的 80 行咭，其他80行咭(包括 IIe)必須拔去。■

列表一：TALK & MUS.PAS

```
PROGRAM turbo_speaker_and_music_editor;
TYPE
     string255  = STRING[255];
     filestring = STRING[14];
{$I HIGRAPH.PRO}
{$I SPEAKER.PRO}
{$I SPEAKER.PAS}
{$I MUSIC.PRO}
{$I MUSIC.PAS}
BEGIN
     CLRSCR;
     inverse;
     GOTOXY(6,2);
     WRITELN('TURBO SPEAKER & MUSIC EDITOR');
     normal;
     WRITELN;  WRITELN('Welcome to TURBO SPEAKER & MUSIC EDITOR')
     WRITELN;  WRITELN('          Hope you will enjoy it !');
     poke_sound_routine;
     play_file('bark.dat');
     GOTOXY(2,4);
     read_talk_routine;
     say_file ('welcome.txt',2),
     REPEAT
          CLRSCR;
          GOTOXY(6,2);
          inverse;
          WRITE('TURBO SPEAKER & MUSIC EDITOR');
          normal;
          GOTOXY(14,4); WRITE('<S>peaker');
          GOTOXY(14,6); WRITE('<M>usic Editor');
          GOTOXY(14,8); WRITE('<Q>uit');
          READ(KBD,ch);
          ch := UPCASE(ch);
          CASE ch OF
               'S'  :   speaker;
               'M'      music_editor,
          END;
     UNTIL ch = 'Q';
END. ■
```

列表二：SPEAKER.PAS

```
PROCEDURE speaker;
TYPE
     string255                      = STRING[255];
VAR
     st , temp_st , filename        : string255;
     loop , temp_mode , mode        : BYTE;
     choise                         : ARRAY['A'..'Z'] OF BYTE;
     item                           : ARRAY[1..5]     OF string255,
     speed                          : BYTE ABSOLUTE $300B;
     pitch                          : BYTE ABSOLUTE $300a;

PROCEDURE point ( ch : CHAR ); FORWARD;

PROCEDURE init;
BEGIN
     CLRSCR;
     choise['P'] := 1;
     choise['S'] := 2;
     choise['F'] := 3;
     choise['C'] := 4;
     choise['Q'] := 5;
     item[1]      := 'PRONOUNCE A WORD'.
     item[2]      := 'SAY A WORD';
     item[3]      := 'SAY TEXT FILE';
     item[4]      := 'CHANGE SPEED OR PITCH';
     item[5]      := 'QUIT';
     speed        := 72;
     pitch        := 64;
     GOTOXY(13,2):
     inverse;
     WRITE('TURBO SPEAKER');
     normal;
     GOTOXY(13,3);
     WRITE('-------------');
     GOTOXY(5,5);  WRITE('< /P >  ',item[1]);
     GOTOXY(5,6);  WRITE('< /S >  ',item[2]);
     GOTOXY(5,7);  WRITE('< /F >  ',item[3]);
     GOTOXY(5,8);  WRITE('< /C >  ',item[4]);
     GOTOXY(5,9);  WRITE('< /Q >  ',item[5]);
     GOTOXY(1,10);  WRITE('Input a string and press RETURN to talk');
     GOTOXY(1,11);  WRITE('---------------------------------------');
     mode          := 1;
     point('S');
END;

PROCEDURE point;
BEGIN
     temp_mode   := mode;
     GOTOXY( 1 , mode + 4 );
     WRITE('    ');
     GOTOXY( 13 , mode + 4 );
     WRITE(item[mode]);
     mode         := choise[ch];
     GOTOXY( 2 , mode + 4 );
     WRITE('-->');
     inverse;
     GOTOXY( 13 , mode + 4 );
     WRITE(item[mode]);
     normal;
END;

PROCEDURE change.
BEGIN
     GOTOXY(1,12).
     WRITELN('       PITCH                     SPEED');
     WRITELN('high  normal  low  /  fast  normal  slow').
     WRITELN(' 20.....64....255      20 .....72....255').
     WRITELN.
     WRITE  ('Current PITCH = ',pitch,'   input PITCH  ').
     READLN (pitch).
     WRITE  ('Current SPEED = ',speed,'   input SPEED  ').
     READLN (speed).
     FOR loop := 0 TO 6 DO
          BEGIN
               GOTOXY(1,12+loop);
               CLREOL;
          END;
     CASE temp_mode OF
          1 : point('P');
          2 : point('S');
          3 : point('F');
     END;
END;

BEGIN
     read_talk_routine;
     init;
     REPEAT
          GOTOXY(1,12).
          WRITE(']');
          READ(st);
          uppercase(st);
          IF st = '' THEN BEGIN
                              st := temp_st.
                              WRITE(temp_st).
                         anD;
          IF (st[1] = '/') AND (st[2] IN ['P','S','F','C','Q'])
             THEN point(st[2])
          ELSE CASE mode OF
                    1 : pronounce(st),
                    2   say(st);
               END.
          CASE mode OF
                    3   BEGIN
                             GOTOXY(2,12);
                             WRITE('Text File to say  '),
                              TADLN(filename);
                             say_file(filename,1).
                             init,
                        END;
                    4 : change;
          END;
          temp_st  := st;
          FOR loop := 0 TO 3 DO
               BEGIN
                    GOTOXY( 1 . 12 + loop ).
                    CLREOL;
               END;
     UNTIL mode = 5;
END; ■
```

列表三：MUSIC.PAS

```
PROCEDURE music_editor;
VAR
     x , y               BYTE.
     index , loop        INTEGER;
     len_val             ARRAY['!'..'5'] OF BYTE.
     pit_val             ARRAY[1..25,1..3] OF BYTE;
     done                BOOLEAN;
     name                STRING[14].

PROCEDURE load_shape( filename  filestring; shape_address  INTEGER).
VAR
     shape_file          FILE OF BYTE.
     loop                INTEGER;
BEGIN
     ASSIGN( shape_file , filename );
     RESET(shape_file).
     loop := shape_address;
     WHILE NOT EOF (shape_file) DO
          BEGIN
               READ( shape_file , MEM[loop]);
               loop := loop + 1;
          END;
     set_shape_address ( shape_address );
END

PROCEDURE init;
VAR
     f                   FILE OF BYTE.
     loop1 , loop2       BYTE.
BEGIN
     load_shape('MUSIC.SHP',$2000).
     len_val['1'] := 16,      len_val['!'] := 24.
     len_val['2'] := 8;       len_val[#34] := 12;
     len_val['3'] := 4;       len_val['#'] := 6;
     len_val['4'] := 2;       len_val['$'] := 3;
     len_val['5'] := 1.
     ASSIGN(f,'music.dat');
     RESET(f);
     FOR loop1 := 1 TO 25 DO
         FOR loop2 := 1 TO 3 DO
             READ(f,pit_val[loop1,loop2]);
     poke_sound_routine;
     y := 11;
     index := 0;
END;

PROCEDURE draw_backgr;
VAR
     loop                : INTEGER;
BEGIN
     hbackground(h_black);
     set_hcolor(3);
     FOR loop := 0 TO 4 DO
          join(0,loop*4+80,279,loop*4+80);
     draw(3,3,74);
     FOR loop := 0 TO 4 DO
          join(0,loop*4+120,279,loop*4+120).
     draw(4,3,136).
END;

FUNCTION real_x(a: BYTE):INTEGER.
BEGIN
     real_x := a * 13 + 10;
END.

FUNCTION real_y(y  BYTE) :INTEGER;
BEGIN
     IF y IN [1..21] THEN
          real_y := (y-1) * 2 + 40
     ELSE
          real_y := (y-1) * 2 + 76;
END.

PROCEDURE print_cursor(x,y :BYTE);
BEGIN
     join(real_x(x),real_y(y),real_x(x)+13,real_y(y)).
END.

PROCEDURE erase_cursor(x,y : BYTE);
BEGIN
     IF (y IN [11..19,22..25]) AND (ODD(y)) THEN set_hcolor(3)
     ELSE set_hcolor(0);
     print_cursor(x,y);
     set_hcolor(3).
END;

PROCEDURE draw_point;
VAR
     a , b                    BYTE;
BEGIN
     a := real_x(x-1);
     b := real_y(y);
     draw(27,a+6,b).
     set_hcolor(0);
     hplot(a+5,b);  hplot(a+9,b).
END;

PROCEDURE cursor_back;
BEGIN
     set_hcolor(0);
     FOR loop := real_x(x)-3 TO real_x(x)+13 DO
          join(loop,20,loop,127);
     set_hcolor(3);
     FOR loop := 0 TO 4 DO
          join(real_x(x)-3,loop*4+60,real_x(x)+13,loop*4+60);
     FOR loop := 0 TO 1 DO
          join(real_x(x)-3,loop*4+120,real_x(x)+13,loop*4+120).
END.

PROCEDURE get_char;
VAR
     temp_y,old_x,old_y : BYTE;

BEGIN
     set_hcolor(6);
     print_cursor(x,y);
     old_x := x;
     old_y := y;
     READ(KBD,ch);
     ch := UPCASE(ch);
     set_hcolor(3);
     IF ch IN ['1'..'5',#33..#36] THEN erase_cursor(x,y);
     set_hcolor(3);
     CASE ch OF
          'I'     y := y - ORD(y>1);
          'M'     y := y + ORD(y<25);
          'J','H'    IF x>1 THEN BEGIN
                                   x := x - 1;
                                   index := index - 1.
                                   cursor_back;
                               END;
          '1','!'   : draw(6,real_x(x),real_y(y));
          '2',#34   : draw(7,real_x(x),real_y(y));
          '3','#'   : draw(8,real_x(x),real_y(y));
          '4','$'   : draw(10,real_x(x),real_y(y));
          '5'       : draw(11,real_x(x),real_y(y));
          'S'       : IF index <> 0 THEN
                        BEGIN
                              pit_ary[index] := pit_val[temp_y,1];
                              IF x > 1 THEN
                                 draw(14,real_x(x-1)-3,real_y(temp_y)-5).
                              sound(len_ary[index],pit_ary[index]);
                        END;
          'F'         IF index <> 0 THEN
                        BEGIN
                              pit_ary[index] := pit_val[temp_y,3];
                              IF x > 1 THEN
                                 draw(15,real_x(x-1)-3,real_y(temp_y)-5);
                              sound(len_ary[index],pit_ary[index]);
                        END;
          'P'         FOR loop := 1 TO index DO
                          sound(len_ary[loop],pit_ary[loop]);
     END;
     IF ch IN ['1'..'5',#33..#36] THEN
        BEGIN
             x               := x + ORD(x<20).
             temp_y          := y.
             index           := index + 1.
             len_ary[index] := len_val[ch];
             pit_ary[index] := pit_val[y,2];
             sound(len_ary[index],pit_ary[index]);
             CASE old_y OF
                  1..9 :  BEGIN
                              FOR loop := 0 TO 4 DO
                                   join(real_x(old_x)-3,loop*4+40.
                                        real_x(old_x)+13,loop*4+40);
                          END.
                  21    : join(real_x(old_x)-3,80,real_x(old_x)+13,80);
             END;
             IF ch IN [#33..#36] THEN draw_point;
         END
       ELSE  erase_cursor(old_x,old_y).
       done := (x=20) OR (ch='Q');
END.

BEGIN
     init.
     REPEAT
          text_mode;
          CLRSCR;
          inverse;
          GOTOXY(12,2).  WRITE('TURBO MUSIC EDIOR');
          normal;
          GOTOXY(8,4).   WRITE('1> Clear memory').
          GOTOXY(8,5);   WRITE('2> Load song');
          GOTOXY(8,6).   WRITE('3> Edit song');
          GOTOXY(8,7);   WRITE('4> Save song');
          GOTOXY(8,8);   WRITE('5> Play song in memory');
          GOTOXY(8,9).   WRITE('6> Quit');
          GOTOXY(10,10); WRITE('Select: ').
          REPEAT
               READ(KBD,ch);
          UNTIL ch IN ['1'..'6'];
          CASE ch OF
               '1' : BEGIN
                          GOTOXY(5,12);
                          WRITE('Erase all the data? (Y/N)').
                          READ(KBD,ch);
                          IF UPCASE(ch) = 'Y' THEN index := 0.
                     END;
               '2' : BEGIN
                          GOTOXY(6,12);
                          WRITE ('File to load : ');
                          READLN(name);
                          load_file(name,index);
                     END;
               '3' : REPEAT
                          hgr_mode;
                          rot := 0.
                          scale := 1;
                          x := 1;
                          ch := ' ';
                          done := FALSE.
                          draw_backgr;
                          WHILE NOT done DO
                               get_char;
                     UNTIL ch = 'Q'.
               '4'   BEGIN
                          GOTOXY(6,12).
                          WRITE('Name of file to save : ');
                          READLN(name).
                          save_file(name,index);
                     END;
               '5' : FOR loop := 1 TO index DO
                         sound(len_ary[loop],pit_ary[loop]).
          END;
     UNTIL ch = '6'.
     text_mode;
END. ■
```

dBASE 中小型工商業應用程式選

信用賬項處理系統(一)

- 周志成 -

在過往數期，筆者為大家詳細地介紹存貨及發票系統。而一套完整的商業處理系統，應是具有多方面功能而能使用相同的資料檔案（可能的話），今次為大家介紹的 dBASE 系統程式是一個名為 PAYBILLS 的信用賬項處理系統，亦是配合貨存及發票系統使用。

筆者過往也曾試過經營一些小生意，在那時期結識了不少朋友，他們經營的行業包括化粧、毛冷、皮鞋及辦館等。雖然他們所經營的行業不相同，但其遇到的經營難題却大致相同，亦是與一般小型商業所遇到的一樣。首先是人手問題，由於要考慮成本，所以僱員的數目較少，通常只聘用2至5人，有時甚至所有工作是自己一人負擔（這情形在香港特別普遍，而商界內俗稱這種公司為皮包公司）。此種經營模式多是把貨物一買一賣中賺取利潤，實際工作量不多，但生意額及貨品種類却絕不可小覷！

信用賬項的產生

一間公司的經營步驟：首先必須適當地理財，前數期為大家介紹的存貨及發票系統便是處理財務運用的其中一部份，今次的信用賬項處理系統則可稱為負債處理系統。大家可能會認為一間健全的公司不應有負債，其實越具潛質的公司，獲信貸的機會越大（信貸非單指銀行方面，是包括供應商所給予的付賬期等），因為對方必然認為你有還款能力才會給予賬期（TERMS），這方面的信心通常來自公司規模、經營年期、經營者信譽及過往業績等等因素。總而言之，你獲信貸越多則反映你經營得宜，而絕非公司不健全的表現，所以信用賬項處理，是一間公司必須具備的理財系統。

賬項與信用的關係

具有信貸能力的公司，可以獲得一些額外的資金幫助發展，無疑是極為有利。但當你的信用賬項越多時，問題也接踵而來，例如公司本身總負債多少？到指定一天要還的款項多少？每項信貸的賬項最後還款期？以上種種問題十分繁複，需要認真地聘專人處理，因為一旦不小心有款項到期而因不察覺未處理，事後除了費唇舌向對方解釋外，公司本身的信用能力可能受到影響，後果十分嚴重。

PAYBILLS 系統

考慮到上文述及的種種賬項問題後，聘專人處理是十分不化算（以小型公司而言），而以上種種問題是十分合適在 dBASE 系統下利用程式處理，PAYBILLS 程式是配合 INVENT 及 INVO-ICE 程式使你的理財系統更趨完善，它的功能十分完備，除了比人手處理效率更高外，出錯機會也比人手處理低得多，祇要你再次收到對方的發票（INVOICE）便把資料輸入，則所有賬項問題及資金運用也在你掌握之中！

PAYBILLS 程式的工作程序與人手處理大致相同，大家在應用時可參考以上步驟：

1. 輸入對方發票的各項資料，如最後付款期等。
2. 每天把公司所有的信貸賬項列出。
3. 選擇那些項目即時付款處理，那些需要先付或押後。
4. 印出所付出金額及供應商資料等。
5. 印製各特別的報表（REPORT）。

上述的步驟詳細來說是輸入供應商的號碼（自己編定）、名稱、地址及最後付款付的日子。至於印製付款紀錄（CHECK）是先經螢幕上的選擇（如那些賬項先付；那些付部份等），然後程式便把今次的付款情況印出，而且把更新的資料記錄回磁碟上，PAYBILLS 系統能助你有效率地處理每次付款的紀錄及更新資料。

程式操作步驟

操作方面，首先你必須處於 dBASE 系統內

圖 1

```
--MAIN PAYABLES MENU--

0-Exit to System

1-Post PAYMENT OF VENDOR Bills
2-Edit Vendor File
3-Post Payables to Checks
4-Print Outstanding Checks
5-Print a Report

Choose A Number::
```

，或直接從 PC-DOS 中鍵入 dBASE PAYBILLS（CP/M 用家也相同），如果已處於 dBASE 系統中則鍵入 DO X：PAYBILLS，而X爲磁碟機編號，程式執行後會首先要求用者鍵入年、月、日的資料，然後便出現如圖一般的畫面，共有五項功能選擇，它們包括：

- 選擇 0 則離開程式及 dBASE 系統回到進入前的作業系統（CP/M 或 PC-DOS）。
- 選擇 1 則可輸入供應商資料（如已存在則可省略）及對方發票的資料。
- 選擇 2 可修改供應商的資料。
- 選擇 3 可以查看每一個供應商及發票資料，顯示內容包括最後付賬日期（DUE DATE）及折扣等，你可參考顯示資料來選擇付欵的先後及數額。
- 選擇 4 則印出此次付欵資料紀錄。
- 選擇 5 則進入報表的副目錄，你共有三種報表可供選擇。

功能一：輸入賬項資料

在主目錄選擇 1，程式會首先要求你輸入 NAME CODE，顯示如下：

What is NAME Code ? : DOE :

當 NAME CODE 輸入後程式會在一個名爲 VENDOR 的 DATA FILE 上搜尋資料，如果能找到相同的 NAME CODE 則會顯示如圖二。

大家可能發覺顯示格式很熟悉，因爲它與 INVOICE 系統的顯示格式相同，項目包括地址、電話等。當資料顯示後，程式詢問你資料是否和你指定的相符，如果按 RETURN 則表示正確，如果按 F 則表示不正確，可重新輸入。

上述的步驟相信大家都十分明白，無論你找尋多少次，必須找到正確的供應商名稱才可繼續下列的步驟，否則要更正便比找尋時麻煩得多。當正確的供應商資料找到後，你可按 T 或 RETURN 便可進入信用賬項輸入模式（圖三），你可看見螢幕上半是供應商的資料，而下半部份則是你需要輸入的項目，首先是 DISCOUNT CODE（折扣率），請大家從左至右觀看，最左一項 DATE DUE 是爲最遲付欵期，接着是 BILL NUMBER（發票編號），此項是助你識別那張發票的資料，而第三項是 DESCRIPTION OF BILL（發票輔助描述），這項你可輸入 20 個字位的文字來描述這張發票的資料，通常是用來輸入貨品及數量的資料，最後一項是金額的輸入。

圖 2

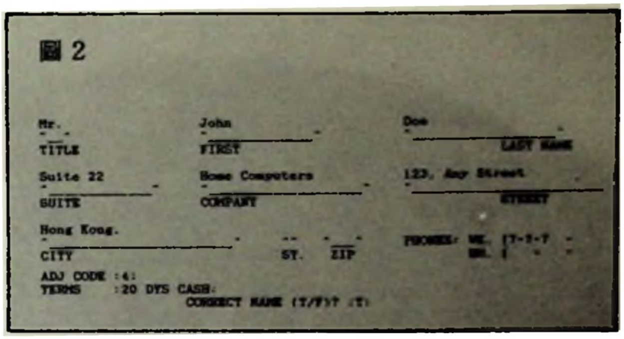


圖 3

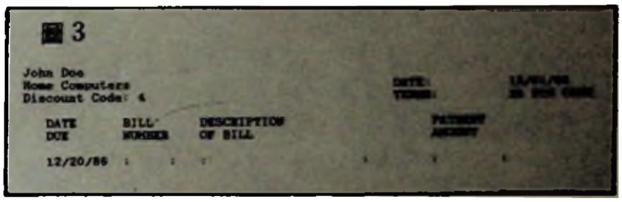
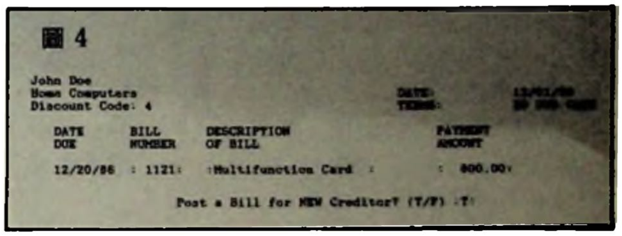


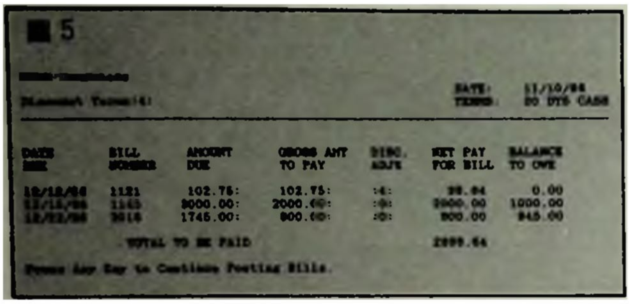
圖 4

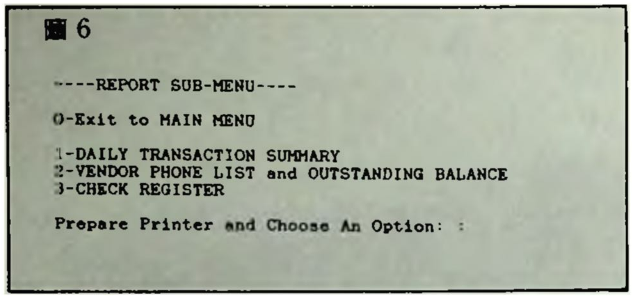


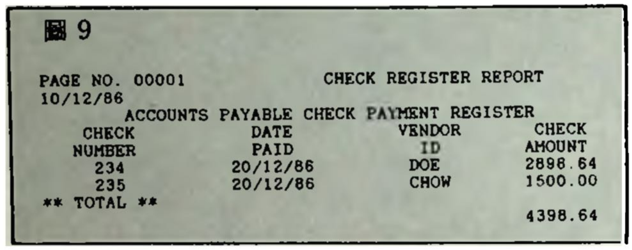
當發票資料全部輸入後，程式詢問你是否還有其他項目輸入（圖四），如果你答T則輸入下一項資料，直至你所有輸入完畢爲止。當所有項目輸入後，程式會詢問你是否繼續輸入其他供應商的信用賬項，如果你答 F 則回到 MAIN MENU。

功能二：修改供應商資料

這項功能是修改供應商的資料，如地址、電話等，詳細步驟大家可參考 V2#8 中 INVOICE 程式修改客戶資料一部份。當資料修改完畢後便會儲回到 VENDOR 這個資料檔案上（DATA FILE）。







功能三：輸入付欵情形

功能三的操作步驟十分重要，因爲此功能是選擇付欵的數目等資料，亦即印出付欵紀錄（CHEC-K）的根據。

程式首先清除畫面，然後是發票（BILL）的資料顯示，你需要輸入的祇是今次付欵數額及折扣率，而這折扣率通常是你和供應商互相協定，而還欵數目則可以分兩或三期形式還欵，當你輸入該兩項資料後，程式會自動地把應付欵項及尙欠欵項。當你把所有應付欵項處理後，程式便會計算出今次付賬總額（圖五），最後你可回到主目錄，再準備印出付欵紀錄單（CHECKS）。

功能四：印出付賬紀錄

功能四的操作十分簡單，其實是把你在功能三時所處理的項目印出，選擇功能四後程式會印出如下句：

****CHECK PRINTING IN PROGRESS ****

有一點大家要十分留意，因爲當付欵紀錄（C-HECKS）印出後，資料便會更新，你便再不能印出第二次，務請小心應用爲上。

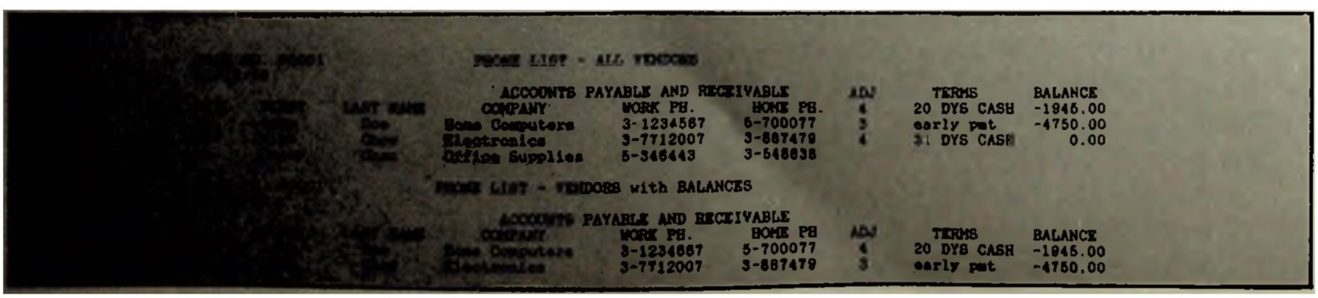
功能五：印出報表

這是最後一項功能，此功能主要是讓你掌握整個信用賬項的狀況，在主目錄選擇5進入報表副目錄（圖六），你共可印出三種報表，而每種報表都有其特別作用，第一項是可以把你指定的日子（例如30，60或90日以上）所應付的賬項列出，這樣你便可以預留一定數目資金以應付賬項，使你可調用多出的資金而又不愁超出預算。第二項報表又可分爲兩種報表，其一可印出所有供應商資料（圖七），而又可祇印出有欠欵的供應商資料，使你可輕易和他們聯絡。最後是印出你已付的發票號碼和金額（圖九）。

結語

PAYBILLS 程式已完整地介紹，還有一些操作上的要點筆者想向讀者說明，這三個財務應用程式 INVENT、INVOICE 及 PAYBILLS 是相輔相承的，讀者應用時最好把它們和 dBASE 系統放進同一磁碟上，這樣做則操作及轉換也容易，如果是應用 IBM 的讀者由於磁碟記憶量較大是不成問題，而 APPLE CP/M 的用家則可能有問題出現，在往後的文章中筆者會教大家如何解決，而下次則會先把 PAYBILLS 程式系統分析及說明如何更改，請讀者留意，再見！

PAYBILLS 系統的各個程式列表將於下期微電腦技術刊出，該期程式磁碟（IBM 及 CP/M 版本）亦會收錄全部檔案。■



拯救WORDSTAR失落的資料

APPLE版本WORDSTAR 3.0/3.3適用

自從 MICROSOFT 公司爲 APPLE II 電腦設計了 Z-80 SOFTCARD 後，APPLE 的用戶便能藉此而運行以往只供 CP/M 操作環境的軟件。CP/M 是一個通用的操作系統，不同機種只要能與 CP/M 兼容的話，便能運行或編寫以 CP/M 爲依歸的軟件。而且 CP/M 能夠有效地管理電腦的記憶以及大容量記憶裝置（磁碟機），使到大型程式能夠以 OVERLAY 形式運行於微型電腦的細小記憶容間。因此不少程式編寫員都致力發展 CP/M 適用的軟件，而出現了有名的 WORDST-AR、dBASE II 等。

CP/M 發展的時候是特別針對科學或工程技術人員，所以並不注重用家友善的方面，不少不明所以的錯誤訊息往往令用家摸不着頭腦。就以在 A-PPLE 上運行 WORDSTAR 爲例，每當你不小心轉換了磁碟、磁碟貼上了 WRITE -PROTECT、有磁區損壞抑或磁頭不緊貼磁碟等情形都會導致程式中斷而回到 CP/M，那麼你一整天工作都會白費了。筆者相信各個經常使用 CP/M 軟件的用戶都會遇過以上的情形，以下則會與大家談談如何在程式中斷後，恢復 WORDSTAR 所失落的資料。

一個典型的 WORDSTAR 文件位於一段連續的 RAM 記憶，實際的位置與你的是 II PLUS 或 IIe 無關，而取決於 CP/M 的版本及 WORD-STAR 的版本。表一顯示不同情形之下你的文件所在的地址，括弧內的數字代表長度（以 256 B-YTES 爲單位）。當文件過長的話，WORDSTA-R 會自動把部份文件存在磁碟，這個時候要恢復的只是在 RAM 的文件，而且因爲要安排文件的儲存位置，表一的地址會略有改變。至於過程要使用 CP/M 系統磁碟上的 DDT，以下便是整個步驟：

①使用一切可能的方法回到 CP/M，試試按 CTRL-C，非必要時不可按 RESET 鍵。

②修正一切錯誤，例如撕去 WRITE-PROTE-CT 等。

③把一張 FORMAT 好的 CP/M 磁碟放在 DRIVE B：。

④把有 DDT.COM 檔案的 CP/M 磁碟放進 DRIVE A，再鍵入"DDT"，爲了方便可以把 D-DT 抄至 WORDSTAR 同一磁碟，當 DDT 載入後可以看到一個"－"號。

⑤鍵入"D846F，CAFF"來把地址 \$846F 至 \$CAFF 的內容以十六進制數字及 ASCII 碼顯示出來。（筆者使用 CP/M 60K 的 WORD-STAR 3.3 作例子，讀者可根據表一作相應的修改。）

⑥把文件捲動至看到結束之位置，按 CTRL-S 作暫停，紀錄這個地址──假設是 \$abcd。跟着可以計算出文件的長度，以頁（256 BYTES）則是 \$mn＝\$ab－\$84＋\$01，可以用 d＝m＊16＋n 來轉換爲十進制。

⑦按兩次"ESC"來終止螢幕的捲動。

⑧鍵入"M846D，abcd，100"來把資料搬移至地址 \$100，以便用 SAVE 指令存在磁碟。

⑨按"CTRL-C"以終止 DDT，回到 CP/-M A＞ 模式。

⑩鍵入"SAVE d B：檔名"，d 是一個十進制數字，由第⑥步所計算出來。

假如你的文件是隔行的話，把第⑩步的 d 改作（10＊N），N 是文件在記憶的頁數；而文件是隔雙行的話，則把 d 改作（5＊N）。另外更快及簡單的方法是儘量把 d 設定爲表一括弧內的值，以把整段記憶存在磁碟。我

我們已討論過整個步驟了，但最好大家先練習一次，以免眞正有需要時手忙腳亂。■

表一 WS Version	CP/M 56K	CP/M 60K
3.3	\$846D-\$BAFF (55)	\$846D-\$CAFF (71)
3.0	\$7849-\$BAFF (67)	\$7849-\$CAFF (83)

1-2-3 及SYMPHONY 的輔助軟件

專業資料分析系統 WHAT'S BEST

一個 LOTUS 1-2-3 或 SYMPHONY 用家可能會遇到下列難題：需要設定一條方程式，並不斷代入不同數字以求找出最大或最小之答案。由於要代入不同數字，所以非常麻煩及浪費時間。不過現在有了這個以線性方程（LINEAR PROGRAM-MING）爲根基的分析資料系統 WHAT'S BES-T，一切問題都可以解決！例如它可以準確地爲一間生產X及Y產品公司顯示出這間公司要生產多少X及多少Y才能賺到最多錢等問題。這些問題通常會出現在經濟學上及商業上。

WHAT'S BEST 版本

WHAT'S BEST 是由 GENERAL OPTIM-IZATION 出品的，共有三種版本：第一種是專業版，須要 640KRAM 才能使用，並可處理4000個變數；第二種是商業版，須要 448K，可處理1500 個變數；而最後一種是個人版，只須 256K，可處理 250 個變數。所有上述的三種版本都可以配合 LOTUS 1-2-3 使用。

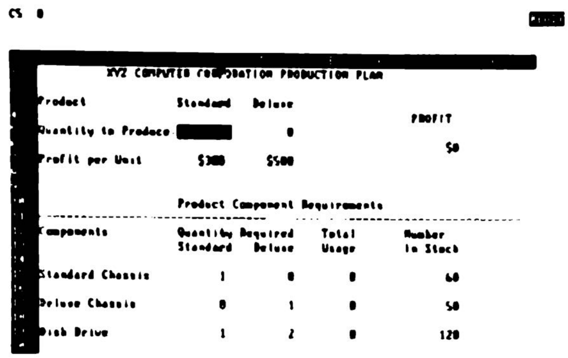
應用例子

爲了使各位更加明白 WHAT'S BEST 的用途，筆者在此特舉一實例以說明：

首先假設 XYZ 公司是生產兩種電腦：一種是標準型配一部磁碟機；另一種是精密型配兩部磁碟機。而現時公司有 60 部標準型電腦、50 部精密型電腦及 120 部磁碟機。問題是該公司應售出多少套標準型及多少精密型才能賺最多錢（由於標準型需一部磁碟機、精密型需兩部，所以 120 部磁碟機是不能全部生產出 60 套標準及 50 套精密型電腦組合的；必須適當分配）。

這類問題要在一般 SPREADSHEET 中解決是非常麻煩的，且找出的答案也未必準確。所以W-HAT'S BEST 便應運而生了！各位可以用傳統方法在任一 SPREADSHEET 下依圖一鍵入資料，然後不斷在 C5 及 D5 輸入數量去計算。這種做法不但花費閣下不少時間及精神，而且得出的答案也未必是最準確的！

圖一



WHAT'S BEST 就是專門用來解決這些問題的軟件；它能巧妙地處理不同變數之組合方式，並得出一最佳答案。但是請留意，若你想處理類似下列的問題：

①當磁碟機售價爲 1000元時利潤是多少？

②售價爲 900 或 1100 元又如何？

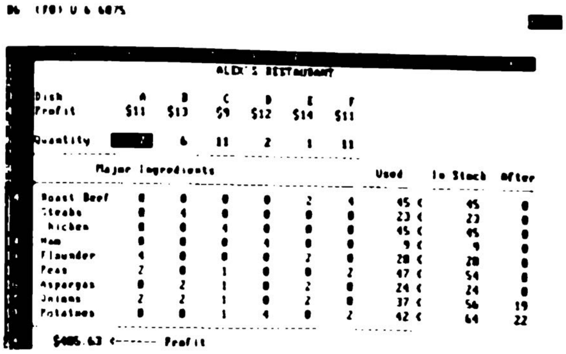
一些諸如此類的問題 WHAT'S BEST 是不能替你解答，不過你可用如 PREDICT 軟件或 LOTU-S 中次數分配函數（FREQUENCY DISTRIBU-TION FUNCTION）去處理。

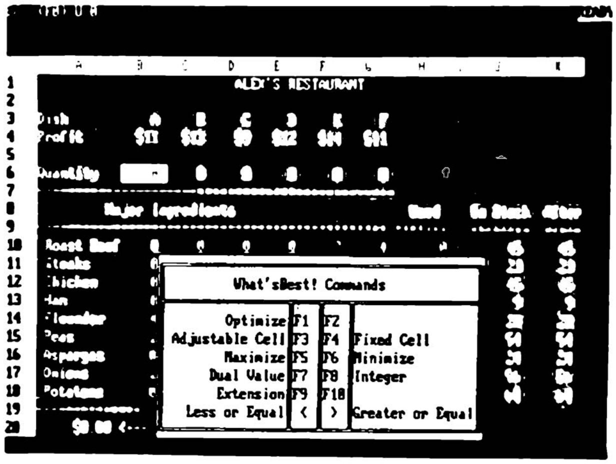
套件的使用

線性方程（LINEAR PROGRAMMING）這個名詞對於一般中五或以上程度的學生應該不會感到陌生！WHAT'S BEST基本上就是使用線性方程原理去解決問題的；而在使用上也是非常容易，你只須學習如何規劃出一 WORKSHEET 即可！

WORKSHEET 中值得一提（即最重要的）的是 ADJUSTABLE、BEST 及 CONSTRAINTS 三項。WHAT'S BEST 便是利用 CONSTRAIN-TS 一項，及不斷運用 ADJUSTABLE 中不同之變數去計算，直至找到最好的一個 RESULT 並將之於 BEST 頁中展示。

專業資料分析系統：WHAT'S BEST





現在讓我們再舉一實例來說明。

假定你是一間飯店的老闆，此飯店有一特色就是逢星期一休息，所以店內之食物必須在星期日全部用光，不能隔夜再用。另外飯店是有六個套餐，分別是 A～F。

現在問題來了，到底飯店要在星期日售出多少 A、B、C、D、E、F 餐，又要能賺到最大利潤；同時也要盡量用光雪櫃之食物呢？

參看圖二，若我們視圖二爲一 LOTUS 之 S-RREADSHHT，解解決上述問題是安土 B6～G PREADSHEET，要解決上述問題是要在 B6～G6 輸入不同的值直至 A20 出現最大值爲止！

相反若我們是用 WHAT'S BEST 來分析上述問題的話，圖二中的 B6～G6 便是 ADJUSTABLE 項；K10～K18 是 CONSTRAINTS；而 A20 即 BEST 區域。K10～K18 就是用 COLUMN H 之值減去 COLUMN J 之值得出的答案。WHAT'S BEST 便是根據這些資料去找出答案的！

此時若你按＜Prtsc＞的話，螢幕中會出現一 MENU（圖三），現將其中幾個較重要的功能簡述如下：

F1 ： OPTIMIZE（後述）

F3 ： ADJUSTABLE CELL，此功能容許你自行

F4 ： FIXED CELL，此功能將使其中一個變數不變，只改變其他變數。

F5 ： MAXIMIZE，用來求出最大值

F6 ： MINIMIZE，用來求最小值

至於其他功能在它的說明文件上有詳細解說。

F1 ： OPTIMIZE 是 WHAT'S BEST 工作之重點所在；當你設定好 WORKSHEET、一切無誤後，可＜F1＞叫 WHAT'S BEST 開始工作。此時它會 SAVE 起 WORKSHEET，脫離 1-2-3，開始分析的工作。而當其工作進行中，是會以 BAR CHART 顯示分析狀況的；一般來講，所需時間約爲 15～20 秒。以上 ALEX 飯店的例子要用 20 秒。在分析完畢後，WHAT'S BEST 會自動返回 LOTUS 1-2-3，載入 WORKSHEET，最後並將結果顯示出來。

總結

事實上，WHAT'S BEST 之應用範圍是非常之廣泛的，而且是研究經濟市場及工商界人士不可缺少的工具。■

CP/M FORMAT 套件軟件 VERSION 2

■用戶可將磁碟容量提高15%　■每套附精裝說明書，只售50元

閣下是否對本頁中所列舉出的文章

感到強烈興趣

現在閣下只要擁有一本「13－24」期合訂本

就可擁有全部



電腦時代13－24期合訂本
現在各特約經銷處購到
每本訂價$60

特輯選
應用軟件庫
實用工具程式篇
最受讀者歡迎評論集
思考性遊戲程式選輯
小型硬件改良／軟件精選
硬件週邊應用研究探討篇
中小型工商業應用軟件庫
中學及兒童教育程式集
微電腦技術85/86精選
高動感遊戲程式選輯
MB應用探討專輯
拆碟技術精華
系統探討篇

本港郵費計算諮詢系統
行政思考決策輔助系統
貸款利息分析計算系統
股票投資分析系統
電腦時代文件櫃
萬用統計圖表製作器
萬能電子計算機
大忙人時間顧問
工商業發貨／帳單系統
日文文字處理系統
被困在迷宮裡的小猩猩
月球一小步．人類一大步
3-6-9門智遊戲
武士忍魂
步步驚營
四位一體
智力鬥的考驗
太空侵略者封閉戰
氫氧碳化學分子排列結構模擬
函數圖展及因子分解
達爾文進化論模擬驗証
英文聽力測驗
東南西北方向概念練習
時間教授
英文字義辨識練習
看圖識字
CATALOG HELLO
VTOC編輯器
將壞磁區自動鎖住的DISK-AID
萬用輸入模組
磁碟LOCKER
程式保留字搜索系統

排序原理透視
活用物理學知識
大力水手POPEYE
可設計路道及車速的死亡大賽車
星球戰士STAR WAR
500個大賽車
吃不到的蟠桃並不是酸的
五子連環
七海爭霸戰
最後一子
讀取磁碟副程式
機械碼輸入編輯器

分析未來開獎走勢之六合彩
16 K WILDCARD
大力水手POPEYE
POKE 214, 128之破解方法
BEAGLE BASIC
有放影功能的FRAME-UP
更趨完美的DOS 3.3
40軌磁碟SECTOR使用情況顯示
40軌磁碟改造器
軟件控制磁碟 BOOT DRIVE 2（下）
如何選擇 BOOT DRIVE 2
將含I/O ERROR的SECTOR除去）

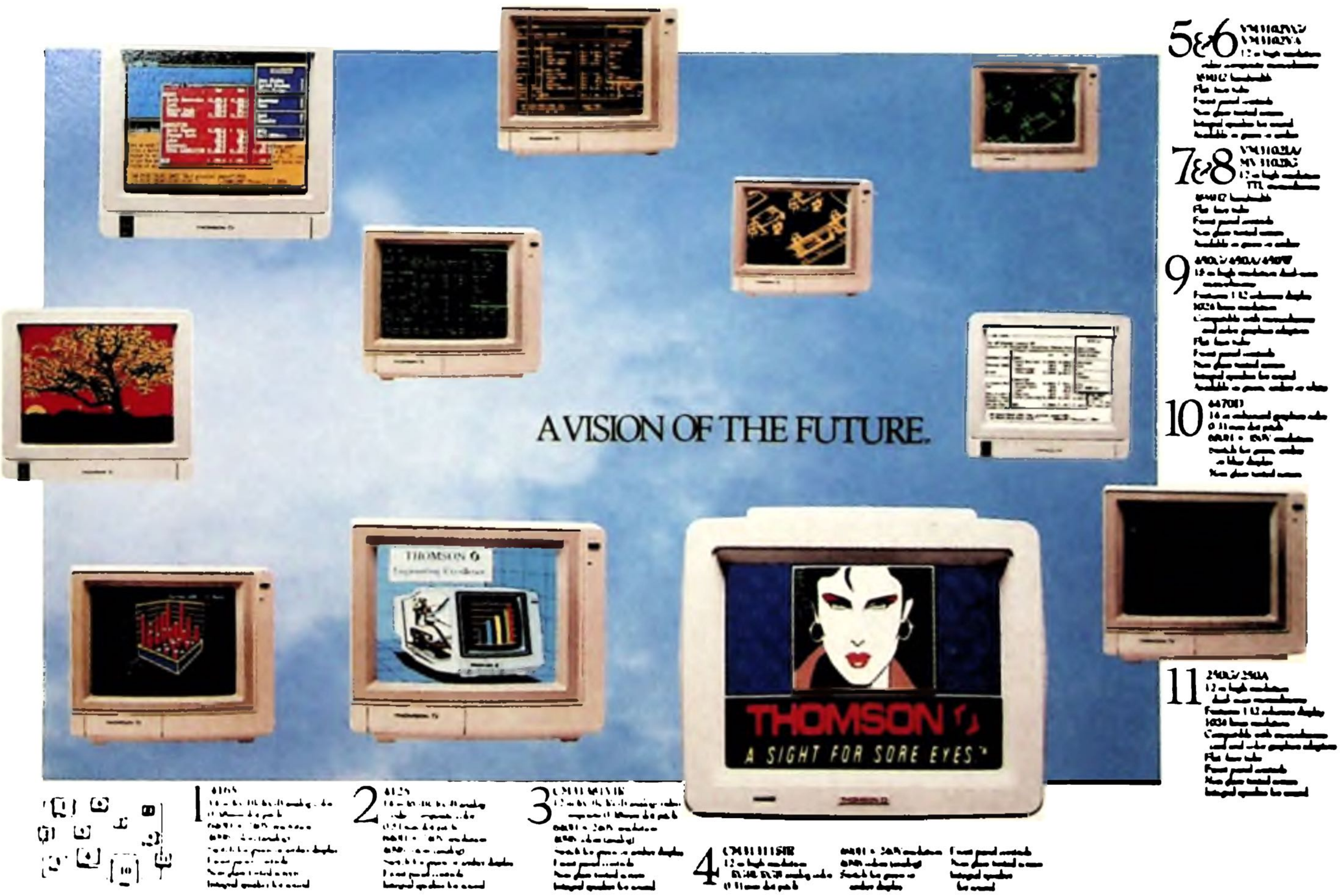
CATALOG HELPER
印製結構性程式列表之LIST大師
APPLESOFT全版編輯器
將視窗顯示功能注入II＋機上
改良版40軌磁碟改造器
輔助程式編輯器
HELLO程式淺談
有七項功能的DOS設計器
APPLESOFT執行監察器
II＋機上的雙倍高解像密度
全天候CTRL字符檢字系統
FORMAT磁碟前警告預報系統　－新INIT指令
機械碼起始地址搬移器
程式編輯器
磁碟保身定造器
專為PS而設計的繪圖工具
將檔案加上保護之檔案密碼鎖
SLIDE SHOW系統
文字畫面螢幕旋轉器
將16K RAM咭模擬為印字機BUFFER咭
功能鍵盤POWER KEY
螢幕顯示格式設計器
迷宮產生原理及如何編寫迷宮遊戲
解決DOS 3.3 SUB-DIRECTORY的方法
機械語言怎樣控制磁碟機馬達活動（上）
機械語言怎樣控制磁碟機馬達活動（下）
APPLE開機後的秘密
印字機功能大揭秘
活用印字機之（一）BIT IMAGE繪印功能的應用
活用印字機之（二）界面咭剖析
一枚卓越設計的CPU－－65C02
APPLE BOOT ROM剖析

THOMSON
COMPUTER MONITORS

NO WONDER WE'RE NO.1

十項全能　極品表現

閣下擁有一部優質電腦，
又怎可缺少一部法國極品MONITOR！！



總代理：世星電腦器材有限公司九龍長沙灣道202-204號瑞星商業大廈5樓A座　電話：3 7283286-8　FAX：3 7850422　TELEX：37017 MSTCO HX

特約經銷商：MICROGRAM COMPUTER COMPANY LIMITED TEL: 3-7282113　MASTER COMPANY TEL: 3-7292956　OCEAN MICROCOMPUTER TEL: 3-864221

微電腦技術程式磁碟

編　號	特價	截止日期
V2#9 APPLE CP/M	$15	1/3/87
V2#9 IBM	$15	1/3/87
V3#1 APPLE CP/M	$15	1/4/87
V3#1 IBM	$15	1/4/87

「微電腦技術」每期刊出之程式列表均有出版磁碟版本，此舉旨在減省讀者鍵入長程式及偵錯之苦。由於本刊內容涉及多款不同機種，故此我們會同時提供APPLE,IBM或其他特別註明之機種版本，大家在選購時請留意。

程式磁碟的售價，由該期的出版日計起九十天內，都會以一個特惠價錢出售。過了九十日的優惠期，程式磁碟的售價會回復原價。讀者可利用郵購方式，或到各特約經銷處購買。

■ V3#1 APPLE CP/M 版本
- 爲 TURBO PASCAL 增加音樂及說話功能
- CP/M 檔案加上密碼保護

■ V3#1 IBM 版本
- 動態記憶管理技巧
- DISKSCAN 磁碟檢查系
- 通訊軟件加上 ANSI 色彩
- 趣味遊戲選之蘋果棋

即期微電腦技術程式磁碟（CP／M及IBM版本）於地下鐵路及九廣鐵路沿線各香港電視服務站均有代售。

省時兼便利 請即利用「電腦時代出版社」提供之訂閱服務

凡電腦時代出版社旗下之出品皆接受訂閱。

訂閱辦法簡易，閣下只需填寫好下列表格及回郵地址，連同訂閱費，利用劃線支票或郵政匯票（請勿劃線）寄來九龍中央郵政信箱 71193 號“電腦時代出版社”收即可。（外埠可用 CPO BOX 71193, KOWLOON CENTRAL POST OFFICE, HONG KONG.即可。）歡迎現時長期訂戶延長訂期。*如要掛號請另加付五元（每期計）。

本人擬訂閱
- ☐「電腦時代」雜誌（十二期），由第____期開始。
- ☐「電腦時代」雜誌＋「程式磁碟」（十二期），由第____期開始。
- ☐「微電腦技術」（　月刊）十二期，由第____期開始。
- ☐ COMPUSOFT 磁碟月刊（十二期）由____期開始。

姓名：________ 地址：________ 電話：________

付上劃線支票No. ________ ________ 原訂戶號碼：________

同時請填好下列之回郵地址：

姓名：________

地址：________

____

郵購訂閱收費一覽表：

種類	香港本地	澳門、中國、東南亞區	其他地區
「電腦時代」雜誌（十二期）	HK$120.00	HK$160.00（平郵） HK$280.00（空郵）	HK$190.00（平郵） HK$340.00（空郵）
「電腦時代」雜誌＋「程式磁碟」（十二期）	HK$475.00	HK$530.00（平郵） HK$690.00（空郵）	HK$570.00（平郵） HK$760.00（空郵）
「微電腦技術」（十二期）	HK$110.00	HK$160.00（平郵） HK$280.00（空郵）	HK$190.00（平郵） HK$340.00（空郵）
「COMPUSOFT 磁碟月刊」（十二期）	HK$120（單碟） HK$360（雙碟）	HK$396.00（平郵） HK$468.00（空郵）	HK$396.00（平郵） HK$468.00（空郵）

沿此黑線剪出

微電腦技術VOL.1/2程式磁碟總目錄

歡迎到讀者服務部補購各期磁碟雜誌

微電腦技術

APPLE/IBM系統，中文系統
網絡通訊專門性電腦技術雜誌

微電腦技術逢每月一號出版・電腦時代逢每月十五號出版

1 JAN.

星期日	星期一	星期二	星期三	星期四	星期五	星期六
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

2 FEB.

星期日	星期一	星期二	星期三	星期四	星期五	星期六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

3 MAR.

星期日	星期一	星期二	星期三	星期四	星期五	星期六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

4 APR.

星期日	星期一	星期二	星期三	星期四	星期五	星期六
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

5 MAY

星期日	星期一	星期二	星期三	星期四	星期五	星期六
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24/31	25	26	27	28	29	30

6 JUN.

星期日	星期一	星期二	星期三	星期四	星期五	星期六
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

7 JUL.

星期日	星期一	星期二	星期三	星期四	星期五	星期六
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

8 AUG.

星期日	星期一	星期二	星期三	星期四	星期五	星期六
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23/30	24/31	25	26	27	28	29

9 SEP.

星期日	星期一	星期二	星期三	星期四	星期五	星期六
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

10 OCT.

星期日	星期一	星期二	星期三	星期四	星期五	星期六
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

11 NOV.

星期日	星期一	星期二	星期三	星期四	星期五	星期六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

12 DEC.

星期日	星期一	星期二	星期三	星期四	星期五	星期六
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

1987

座枱年曆製作方法

①依左面黑線將本頁從書中剪出
②照虛線所示位置對摺，先用刀片輕剔則效果更佳
③把本頁兩端貼上膠紙，即成一精美座枱年曆

微電腦技術 新讀者需知

微電腦技術是一本專門性的電腦技術雜誌，對象是較高層次的電腦用戶。而每期的內容都是環繞硬件方面的最新發展、軟件編寫技巧和應用，以及資料性的報導等。本刊所針對的機種包括了最爲流行的 APPLE II 及 IBM PC/XT/AT，務求把讀者帶至更高的水平。

APPLE CP／M 系統

祇要在 APPLE II 內插上一張 MICROSO-FT Z80 SOFTCARD 或是 APPLIED ENG-INEERING 的 Z80 PLUS 便可用 APPLE 來運行大量 CP/M 軟件。供 APPLE 使用的 C-P/M 系統計有 44K、56K、60K 及 64K，其中 56K、60K 及 64K 的 CP/M 均要插上 16K（或 128K）RAM 咭才可以，但由於大部份的 CP/M 軟件都需要 56K 或以上的記憶，故我們建議大家購買一張記憶擴展咭。另外，一張插在 SLOT 3 的 80 行咭亦是運行的必需品。

雖然 CP/M 的 SYSTEM MASTER 已附有很多的工具軟件，但我們向大家推薦一個優良的工具軟件 —— POWER，其抄錄檔案功能將使 CP/-M 用家更覺方便不少。

IBM 系統

應用於 IBM PC/XT/AT 的操作系統有很多，其中包括 CP/M-86、PC-DOS、XENIX 等。不過始終是以 PC-DOS 較爲普遍，現時大家可以找到最新版本的 PC-DOS 是 3.1，此版本可以支持使用大容量的磁碟（1.2MB，XT 用家需要配合特別之磁碟機控制咭及大容量磁碟機）。PC 的標準記憶爲 64K 而 XT 則爲 256K，爲了照顧軟件的需要，加上一張 384K 的 MULTI-FUNCTION 咭可使整個系統的功能更加完善。

爲了使處理檔案更加方便，我們建議大家使用 PC-TOOLS（由 CENTRAL POINT SOFTW-ARE 所出版），無論抄錄檔案或 FORMAT 磁碟都簡單得多。

dBASE 應用

dBASE 是一個十分出色的資料庫管理系統，很多方面都勝任愉快。在 IBM 上運行的 dBASE 共有三個版本 —— II／III／III PLUS，而 AP-PLE CP/M 系統就只能運行 dBASE II。至於 dBASE II 和 III 最大的分別是 II 最多每次只能開啓二個檔案，而 III 則是十個，III P-LUS 則是加上了多個用戶操作能力以及增強了的指令。APPLE CP/M 的 dBASE II 和 IB-M 版本的 dBASE II 亦有差異，例如 APPLE CP/M 的 dBASE 程式檔是 .CMD，而 IBM 則是 .PRG，具體情形請參考說明手冊。

dBASE 最大的優點是擁有編寫程式[illegible]本刊每期都刊登了不少的 dBASE 應用[illegible]些程式大部份都能適用於 APPLE CP/[illegible] B-M 系統。要執行這些程式，可把 dBAS[illegible]磁碟機 A，而把微電腦技術程式磁碟放在磁[illegible] B，進入了 dBASE 後鍵入 SET DEFAULT TO B，再 DO 檔案名便可以了。

PASCAL 程式技巧

PASCAL 是一種結構化的程式語言，一般可以接觸得到的有 APPLE（UCSD）PASCAL 及 TURBO PASCAL，供 APPLE 使用 TURBO PASCAL 的爲 2.0 版本，而 IBM 版本則有 2.0 及 3.0。概括而言 APPLE CP/M 的 TU-RBO PASCAL 與 IBM 的 TURBO PASCA-L 是相容的，只有使用到特別的硬件（記憶、畫面）的時候才有分別。

程式磁碟

爲了省却大家鍵入冗長的程式列表以及除錯的麻煩，每期微電腦技術所刊登的各個程式都會收錄在同時出版之程式磁碟內。由於部份程式只適用於某一機種，我們分別出版了 IBM 及 APPLE C-P/M 版本的磁碟，程式磁碟可以到各特約經銷處或讀者服務部購買。

微電腦技術程式磁碟並沒有放進操作系統，故不能使用作 BOOT 機。如要觀看程式磁碟之內容，應該先放進系統磁碟（例如 CP/M 的 SYST-EM MASTER 或 PC-DOS 2.0 以上版本），再鍵入 DIR 便可。至於各個程式的執行步驟，請翻閱內文之介紹。■

電腦時代
又一貢獻

蘋果中文咭應用程式專集

經已出版

蘋果中文咭應用程式專集是一本任何擁有中文咭的用家不可缺少的參考工具書。本書除了收集有多個實用性極高的程式外，更有多篇極有深度的文章探討 CCDOS 的秘密及編寫中文咭應用程式的技巧。全書厚 64 頁，每本只售 25 元



以下是本書部份精彩內容介紹：

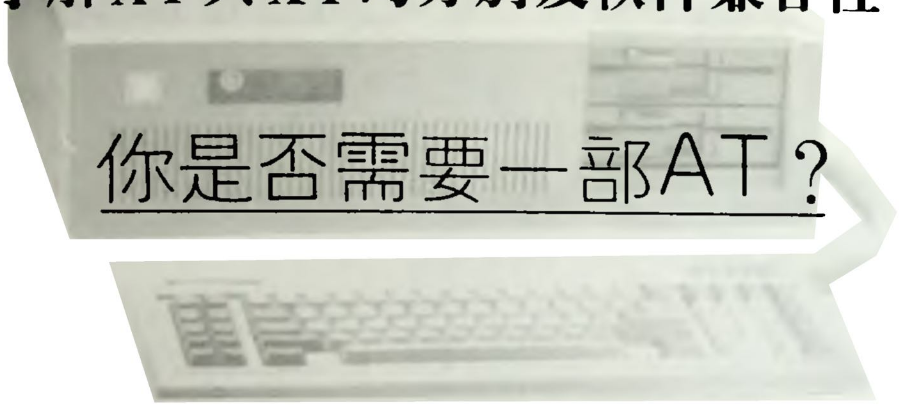
- 爲中文咭加添造字功能
  現時的中文咭是沒有造字功能，本程式針對這點，令使用者能自行設計新的字彙或圖案，新字能在任何的程式中使用。
- 超濃縮的 CCDOS＋
  CCDOS 佔用了磁碟的 106 個 SECTOR 空間實在浪費，新的超濃縮的 CCDOS＋只佔用廿餘個 SECTOR，節省五份四的佔用空間，而修改顯示頁或其他任何參考數時更加方便容易。
- 首個中文咭程式編輯器：D.BUG（CHINESE）對於喜愛編寫中文咭程式的朋友，現時是沒有任何能在 CCDOS 系統下使用的編輯器，D.BUG（CHINESE）將會提供大部份你希望擁有的編輯功能。
- 中文大字編印器
  本大字編印器不單能隨意編印大小、濶窄比例不同的大中文字外，更能隨意將字作 90 度、180 度及 270 度旋轉，而且更可以印出反白大字，效果超乎想像。本程式更可配合中文咭造字程式使用。
- 倉頡中文輸入法練習
  一個由淺入深的練習課程，使用家能對倉頡中文輸入法更加有認識及掌握。
- 六合彩分析及預測開彩結果
  分析六合彩各期號碼的走勢及預測下期開彩的結果，程式簡單易用。

特約經銷處

本刊經已出版，大家可在下列特約經銷處購買：

1. 港九地鐵／新界九廣鐵路各電視服務站
2. 黃金商場地庫 21 號萬達電腦公司
3. 中環三聯書局（中央街市側）
4. 西環屈地街 22 號地下創業雜誌中心
5. 灣仔洛克道 54 號三樓 301 室世紀電腦軟件公司

詳細了解AT與XT的分別及軟件兼容性



你是否需要一部AT？

現時流行的 IBM PC AT 基本上有兩款結構，基本設計（3995 美元）爲 256K Bytes 的隨機存取記憶（RAM），一個新設計的高容量 1.2m-egabyte 磁碟機，及一張合併了軟磁碟／硬磁碟的控制咭。另加 1800 美元可得到額外的 256K-bytes 記憶的增強設計，一個 20-megabyte 的硬磁碟機，及一張串聯／平行介面接合器。這兩個系統都是以 Intel 80286 處理器操作，具備 8 個 I/O（輸入／輸出）擴展槽，及一個有後備電池的鐘／日曆。

AT 附送了多份說明書，包括一裝置指南，一本操作指南，及一本 BASIC 手冊，全部都放在 IBM 的標準活頁格式盒內。

BASIC 手冊已非常完整，閣下無需把附頁放入，便可以代替原來之頁數取得文件之校正（IBM 於去年中曾再版發行）。

電源供應及鍵盤

AT 的電源供應爲 190WATTS，相對地 PC 僅 63WATTS，而 XT 則有 130WATTS。PC 的功率是不足夠的，當需要加入附件於系統，亦要較大的功率操縱兩部硬磁碟機。由於發出的熱量頗大，IBM 使用新發明的變速風扇，當內部氣溫昇高時它轉動較快（更響）。負荷低時，噪音並不打擾情緒。一項值得注意的改良是 AT 有電壓選擇制，使它能操作於歐洲及香港的 220 伏特電壓。

AT 的鍵盤及介面與 PC 並不兼容，AT 的鍵盤不能用於 PC 上。一個單片微處理器在系統主板上負責管理鍵盤，及有關功能。任何直接經鍵盤介面硬件的 PC 軟件，一些字鍵翻譯程式，及大部份遊戲均不能工作於 AT。

鍵盤佈局類似 IBM 電動打字機。SHIFT，CONTROL，ENTER，及 BACKSPACE 鍵全部被放大。一些少用的鍵，例如“＼”，“～”，PRINT SCREEN，及 ESCAPE 被移往鍵盤的邊緣部份。

三支狀態燈被加在 Caps Lock，Scroll Lock，及 Num Lock —— 這是受歡迎的特色。唯一新字鍵，SYS REQ，無論字鍵按下或放開，都會導致 ROM（只讀記憶）內的鍵盤操縱軟件產生一軟件插斷，讓用家給訊號操作系統，引起它的注意。不過 PC-DOS 目前忽略 SYS-REQ 鍵。

與國際電源供應同時推出的是 IBM 爲外國語言提供了六款不同版本的 AT 鍵盤，它的佈局及內建掃描碼全部一樣，然而爲著允許使用獨特的語言符號，字鍵的插圖說明有所不同。標準的顯示接合器能夠顯示這些字符，同時 DOS 3.0 有一組應用程式可把自己改編爲指定的鍵盤類型。

輸出方面，AT 採用標準 PC 顯示咭，因此絕對兼容。另一方面，圖像產生較 PC 快得多。

AT 主機裝有鎖匙開關，它可使鍵盤斷路及鎖著機蓋，這特色似乎作用不大。因爲閣下需保護整個系統，及外接引線避免某些人意外干擾執行中之程式。有一程式能夠測試鎖的狀態，及不顧它的功能去選擇由鍵盤取得輸入。

系統主板

系統主板本身是一個全新設計，它以 Intel 的高性能 80286 微處理器取代 PC 的 8088。一個空的插座提供給 80287 數字處理器，加快浮點數值之運算速度。主板除插有部份熟悉的零件外，很多是新設計的。

主機開動時，80286 是以眞正位址模式（Real Address Mode）操作，同時使用上與 PC 及 XT 相似。好像 8088 一樣，它使用分割位址體制來存取高達 1megabyte 的記憶，它有

你是否需要一部AT？

相同的指令組及幾個擴展與 8088 不相容的指令。

最主要的分別是 80286 執行速度更快；它使用較快的時鐘（6MHz 對 4.77MHz），以 16-bit 數據線（DATA BUS）取代 8-bit 數據線。然而，增加速度的主要原因是來自內部的改良，它僅以 8088 需要的一半時鐘周期執行大部份的指令。最後的結果是，當運行集中計算的程式時速度比 PC 或 XT 快 4 至 6 倍。

當 80286 進入它的保護位址模式（Protected address mode）時，情況更爲有趣。雖然它依然執行同樣基本的指令組，它的操作更接近地類似大型電腦的 VIRTUAL MEMORY，特別適合多用途及多用戶應用。

儘管 80286 在它的 68-pin 外殼包裝得很好，它並非最好的微處理器。某些類型的功能它可執行得出色，好似評價有效的實質記憶及快速地爲實時應用工作開關，然而却存著缺點。像 8088 一樣，80286 的主要問題是集中在使用中的分割區域，因爲每區的大小限制爲 64Kbytes，對付巨大的陣列如圖象及訊號處理時便顯得吃力。另一微處理器 Motorola 的 68000 擁有巨大的線性位址空間，通常較爲有效。

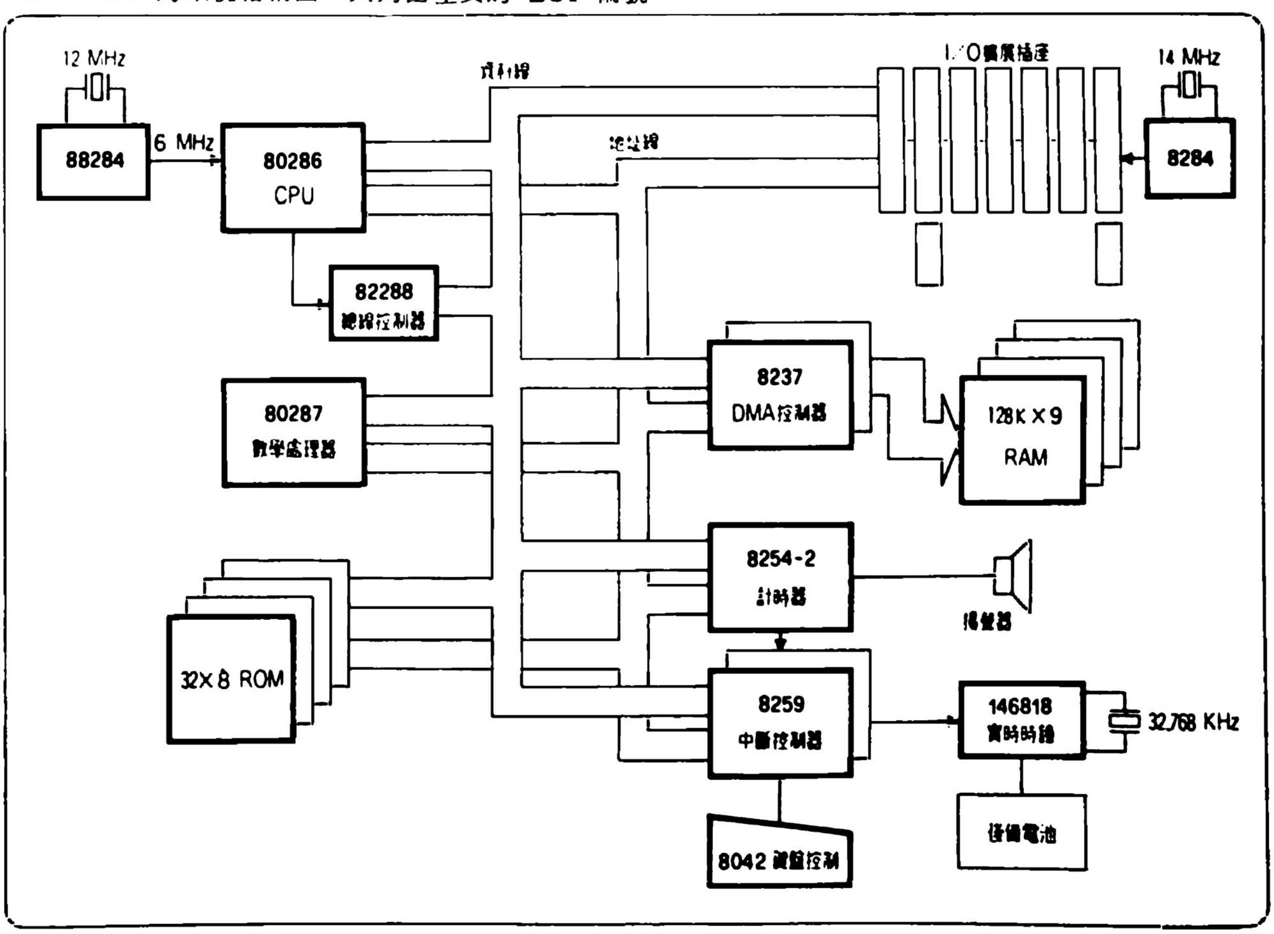
軟件的兼容性是另一難題，程式爲眞正模式編寫通常不能於保護模式執行，相反亦一樣。應用程式只需些微修改（通常只需重新組合），然而閣下不能將現存的軟件運行於 80286 的保護模式，系統軟件對微處理器的架構會要求得更嚴謹。PC-DOS 只可工作於眞正模式，就算 IBM 本身的 ROM BIOS（基本輸入／輸出系統）進入保護模式後也變成沒用。Microsoft 的 XENIX 是唯一的操作系統聲稱可使用保護模式操作。

AT 支持了 80287 數學處理器，售價爲 375 美元。儘管這改動並不是特別大，80287 並非與用在 8087 的軟件完全兼容，因此以 8087 編寫之程式未必可在 AT 內工作，實際增加的效能是只取決於應用軟件。

系統主板留有供 512Kbytes 同等測試 RAM 之空間。AT 有 256Kbytes，而增強設計有 512Kbytes。

在系統主板上有 8 條全長的 I/O 槽，較 XT 的 6 條全長及 2 條短槽提供更多擴展的能力。而且軟磁碟及硬磁碟功能合併在一張咭內可騰空出一個

圖一：AT 的系統結構圖，只列出主要的 LSI 編號



附加擴展槽。

每一條槽都是裝置有平常 62-pin 接合器，雖然計時並不相同，這些接合器與 PC 一樣載著相同的訊號，這些槽中其中六條附有 36-pin 接合器爲 AT 的特殊咭而設，並且包含伸延位址線（A20－A23）讓閣下在系統內放入高達 16megabytes 的記憶。數據線的高 8 位元亦位於此。要容納現有的 8 位元 I/O 咭，系統主板上的硬件自動地把每個微處理器初設的 16 位元數據或 I/O 傳送轉換成兩個 8 位元傳送。任何支持 16 位元傳送的咭都能經這接合器送返訊號把轉移失效。

一個有趣的訊號"Master"，讓 I/O 咭上的微處理器暫時取代系統的控制權及存取任何記憶或外圍裝置。這能力爲有智能的外圍裝置及連同微處理器的咭開啓新的機會。

與 PC 的 I/O 咭雖有良好的兼容性，然而却非百分之一百。高速的系統時鐘引致要放棄很多咭而不能於 AT 上操作，PC 的記憶擴展及多功能咭都不能工作。另一方面，AT 加入的記憶及時鐘令主板有些多餘，而超過 1megabyte 的擴展咭可由 IBM 及其他發展商購買。列表一列出 IBM PC/AT 支持那些擴展選擇。

有幾張咭，如 IBM 的彩色圖象咭，不能插入雙接合槽內，因爲它們伸延低於接合器頂部。這些咭一定要被放置在單接合槽其中之一。因爲機殼比 PC 的高，爲 AT 設計的咭可增高約一吋。

龐大儲存量

AT 是第一部使用新一代高容量軟磁碟機的個人電腦，這些磁碟機有能力把 1.2megabytes 的記憶放在特製的 5-1/4 吋磁碟。數據被放在 160 軌（每面 80 軌），而每磁軌有 15 個 512byte 的磁區。數據傳送率爲 500Kbps，大於標準磁碟約兩倍，轉速由原來的每分鐘 300 轉增加至 360 轉。

要取得如此高密度，一定要使用高質素磁碟（2HD）。因爲各位元塞得接近一起，多到每吋 10,000 位元，用來寫入數據的磁場很易影響鄰近的位元。磁性記錄介質要求更強的磁場去固定或強制一位元，它忽略較弱的雜散磁場，並且只受記錄頭下面的強磁場直接影響。

IBM 發展一張新的磁碟接合咭應付這類磁碟機，亦適用於標準的軟及硬磁碟控制。不像舊的磁碟接合器，這張新咭只可應付兩部軟磁碟機。然而，倘若能夠找到操作軟件及有地方放置更多磁碟機，是可以使用兩張控制咭。

高容量的磁碟機及控制咭能讀取標準之磁碟，原因是目前沒有軟件應用於高容量格式內。閣下亦可寫入標準之磁碟，但因爲記錄之磁軌較窄，因不能用標準之磁碟機讀取。這意味著 AT 用家如需要由 PC 傳送數據會被迫犧牲一部高容量磁碟機，而使用標準磁碟機，或者使用串聯通訊掛鈎。

不斷試用此高容量磁碟機後，未見任何數據錯誤，或者就算重新使用與系統一起推出之特殊 IBM 磁碟也沒有問題。

ROM 內的 BIOS 在磁碟機重置後自動地決定磁碟機／格式組合；這令到眞實控制器機件透徹一程式，它亦令到很多拷貝保護技巧不能兼容。令人費解是 BIOS 強迫磁碟系統最少要有八分一秒摩打啓動延遲時間。引致每次最初之磁碟存取較 PC 更久，半高度之磁碟機啓動時間較長，然而筆者仍望這參數保持爲一變數，以方便不同情況下進行修改。

倘若閣下需要硬磁碟機，增強的 AT 有 20megabyte，全高度，放入機殼內的硬磁碟機（表二列出三款硬磁碟操作之時間比較），可加多第二個硬磁碟取代另一軟磁碟機。

ROM 內的 BIOS

ROM 內載有卡式 BASIC 翻譯器（AT 沒有

圖二：AT 的系統記憶分配

000000	- 07FFFF	512 KB	System RAM
080000	- 09FFFF	128 KB	Option expansion board
0A0000	- 0BFFFF	Video RAM	Graphics display adapter
0C0000	- 0DFFFF	128 KB	I/O Expansion ROM option
0E0000	- 0EFFFF	64 KB res.	Duplicate assignment for FE0000
0F0000	- 0FFFFF	64 KB res.	Duplicate assignment for FF0000
100000	- FDFFFF	Max. 15 MB	I/O Memory expansion
FE0000	- FEFFFF	64 KB res.	Duplicate assignment for 0E0000
FF0000	- FFFFFF	64 KB res. (ROM)	Duplicate assignment for 0F0000

列表一：AT 適用的擴展選擇

SUPPORTED	NOT SUPPORTED
IBM monochrome display adapter	IBM asynchronous communications adapter
IBM color display adapter	IBM printer adapter
IBM SDLC communications adapter	IBM expansion unit
IBM binary synchronous communications adapter	IBM compact printer
IBM cluster network adapter	Other memory expansion options
IBM PC network adapter	Other keyboards
IBM graphics printer	Other disk and fixed-disk drives
IBM color printer	

列表二：三款硬磁碟機的操作速度比較

TEST	IBM PC AT	IBM PC XT	APPLE PROFILE
BASIC			
Hard-disk Write	17	43	22
Hard-disk Read	12	28	13
SYSTEM UTILITIES			
40K File Copy	12	2.8	20
SPREADSHEET			
Load	1.7	3.8	N/A

卡式介面），一開機自我測試程式，及在四個 16K 乘 8 bit 裝置內的 BIOS 功能。倘若閣下移去跳線，一對 32K 乘 8 bit ROM 亦可做同樣工作，並空出插座給擴展用。與 PC 一樣，擴展R-OM 可被 ROM 內之 BIOS 認出並與它的功能結合。

AT 有一新版本的 BIOS，它提供幾種新的特色。最值得注意的地方爲加入支持多用戶操作系統，目前只有少部份 PC 操作系統能夠同時執行多過一個程式，DIGITAL RESEARCH 的 Concu-rrent DOS 及大批的 UNIX 基本套裝是較爲出名的。當閣下呼叫它開始一件操作（例如存取磁碟），直至 BIOS 完成爲止閣下不能做其它工作—— 就算微處理器利用大部份時間等候。這是爲甚麼當 PC-DOS 正在傾印畫面而一磁碟需要存取時，鍵盤輸入好像進入間歇性停頓。

然而，AT 的 BIOS 功能可以回覆一旗號予呼喚程式代表 80286 非常繁忙。之後操作系統在硬件爲這而工作時，執行另一程式。當操作完成時，BIOS 設定另一旗號代表已完成及軟件返回原來之程式。

儘管這特色很有幫助，它（及一般的 ROM B-IOS）只有眞正模式才可執行。可能例外的多用戶便利建設在 IBM 的 TopView 內，新的多用戶系統在實際模式內操作並包括它們本身的 BIO-S。

另外新的特色已設計成由硬件隔離程式供以後及以前之軟件兼容，這些包括遊戲捧支援及一短中斷（微秒）計時器。

AT 有一個潛著的有用新功能，但却有些隱藏著的問題，原因是 PC-DOS 僅支持首 640Kbyt-es 的記憶，IBM 在 BIOS 內建立一功能，允許標準及包括使用伸延記憶與磁碟的空間以區域傳送，達到虛擬記憶性能。

BIOS 的 Move Block 功能就是用作這個用途，操作方法非常簡單：只要將微處理放在保護模式做傳送，並再次返回眞正模式。問題是由保護模式返回眞正模式的唯一路途會重置微處理器。不過最初在後備電池結構 RAM 內設有旗號給訊號指出，而這重置是因爲這特殊理由。接近初始化程序開頭之時，旗號被查出並且程式再次返回 Move Block 改正。

使用這方法有兩重點，首先整個操作要4或5微秒，所以一定要關閉所有挿斷才可工作。這樣可以延遲集中挿斷於這點操作，免於這點有危急的事件發生。例如在 9600-bps 的串連埠可能錯過字元。另一問題較為嚴重，倘若在細小時間視窗電源或系統故障，而期間旗號已被設立，則每次系統開啓或重置時，它會想像是由 Move Block 返回並失去控制。要系統再次工作的唯一方法是打開主機，並且拆開電池一會把旗號清除。閣下亦需重置時鐘及再執行結構程式。因此最好還是採用更便宜及可靠的做法是堅持使用硬磁碟作快速貯存。

新的PC-DOS

與 AT 同時推出的 DOS 3.X 版本是 PC-DOS 的新版本，它符合兩項目的。首先，它提供在 AT 上執行時所需之內部改動。其次亦供給間歇性的釋放，讓程式員開始把他們的軟件與 IBM 宣稱於 AT 局部區域網絡環境操作之工具接合，使到檔案可以由各用戶分享。

檔案分享在多用戶或網路系統內被要求保証同一時間只有一個用戶能夠修改或記錄一個檔案，不然，修改或更新可能不會被正確記錄。雖然局部區域網路用在 PC 已有一段時間，它們各有不同的分享方法，軟件發展者也不願為各種網路建設不同的版本。雖然直至 DOS 3.1 出現前實際的網路軟件不能得到，DOS 3.X 為發展者訂定軟件接合的標準。

DOS 3.X 修改了 DOX 2.X 內幾個次要的錯誤，並加入一些新指令。最令人欣喜的修改是可在指令前使用路徑名稱，同時修正 FOR 這個批次指令，在此之前不可能處理超越 64 字元長度。新指令的功能包括支持外國語言的鍵盤，令檔案為僅讀，及在磁碟內改變標記等。

主要的內部改動是讓 PC-DOS 處理由原來的 4086 增為 65,526 個磁碟分配區，這樣允許在更大的硬磁碟機更有效地使用磁碟空間。

BASICA 亦被增強，但是修改實在是對提供文件而言。有幾個關鍵字如 SHELL ENVIRON$，及 IOCTL 被保留但未被定義，最後以指令及函數被包括在手冊內。儘管以前的版本大部份指令存有錯誤，但這版本只簡略地承認它們。

假如不依規則操作，對系統內某些次要的修改可能對程式導致問題。其中一項修改是全部檔名字元內 8 位元使用來支持外國字符組，令到 Digital Research 的 GSX 圖象伸延版本不能用在 AT 及 PC，不過最新的版本已解決此問題。

總括

除個別獨立程式及 GSX 外，所有程式均正常運行。表三列出那些 AT 可以工作正常，那些不能。IBM 聯同 AT 一起供應一本小册子告訴閣下那些程式它的公司知道不能運作，並為支持的軟件提出任何需要特殊顧慮的因素。最重要是包括如何拷貝程式往高密度磁碟的指示。

IBM 列明幾種程式不能在這些磁碟運作，原因是拷貝保護技術或假設程式有關磁碟佈局。閣下要在標準磁碟機上運行。

總結上文所講，IBM PC AT 在執行及貯存速度增加4至6倍，可用來取代 PC 或 XT 系統的強勁機器，體積細小，價錢合理，多用戶商業系統。筆者期待 XENIX 與此兼容，使日後的 AT 發揮強勁的功能。■

表三：兼容於 AT 的軟件

COMPATABLE	NOT COMPATABLE
[illegible]MATE Editor	Flight Simulator
Ci-C86 C Compiler	J-Bird
Lotus 1-2-3	Frogger
Wordstar	Burgertime
Multimate	PC-Man
[illegible]Write	CP/M-86
Multiplan	Concurrent CP/M
SuperCalc	DR's GSX
[illegible]Text	
[illegible]	
[illegible] II	



一個高級的多用戶多工操作系統

CONCURRENT PC-DOS

一個以電腦作爲休閒娛樂的人來說，他根本不需要考慮到更換作業系統(Operating System)的問題。可是，對於利用電腦作其他商業或實際用途的人來說，這却不是一個易於解決的問題。

IBM PC 的用戶可以使用的作業系統非常之多，大部份的程式都可以在不同的作業系統上運作，但對於一些有特定要求的程式，却需要有特殊的作業系統始能執行。

最近，Digital Research 公司又推出一個新的作業系統，使得 IBM 的使用者又有多一個不同的選擇。

Digital Research 公司始創了全世界最普遍採用的八位元電腦作業系統：CP/M。這個作業系統的十六位元電腦版本：CP/M-86，於 1981 年被引進而用於 IBM PC 中。CP/M-86 和跟着推出的 Concurrent CP/M-86(CCP/M-86)，以及有視窗的 CCP/M-86，都沒有像 CP/M 於八位元電腦中一樣，取得應有的成功。因爲在此同時，Microsoft 推出的 MS-DOS，亦已迅速成長，且日益大衆化，使用的人也愈來愈多了。

該公司最近所推出的新產品，是 CP/M 家族的成員，名爲 Concurrent PC-DOS 3.2。這個新產品對於一些有特殊需要的 MS-DOS 使用者來說，將會是一個非常適合的作業系統。

Concurrent PC-DOS 具有五項非常明顯的特點。首先，本作業系統可以使用最多四個視窗進行多工作處理，當然，多工作處理早已在 CCP/M-86 中出現，但 MS-DOS 却沒有此特點。其次，Concurrent PC-DOS 具有多使用者處理能力，可供最多三個使用者同時使用同一部 PC。第三，除了某些限制之外，利用 Concurrent PC-DOS 去執行 MS-DOS 寫成的程式時，不必更改源程式(source program)，不必重新編譯，亦不必增加硬體設備。第四點，Concurrent PC-DOS 可以裝置(Install)在一個以 CCP/M-86 和 MS-DOS FORMAT，並分別具有這兩種型態的應用程式的硬磁碟機上。

最後一項是市場上的革新——也就是 Concurrent PC-DOS 的名稱。原本是 MS-DOS的 IBM 版本的 PC-DOS，這個名稱從來沒有受到法律的保護，但現在 Digital Research 宣稱已擁有了這名稱的專利權。爲了避免混淆，在本文中，將以"IBM-DOS"來表示 IBM 的版本，而"C/PC-DOS"則代表 Concurrent PC-DOS。

Digital Research 發展 C/PC-DOS 的目標，是要闖入三個不同的消費群市場。

第一個消費群是擁有 IBM PC/XT 的用戶，而且通常都使用兩個或以上的應用程式，例如 Lotus 1-2-3，Wordstar，dBASE II，或是 PFS 檔案等。在這個消費群中最典型的使用者是財務分析師或業務顧問。他們需要在不同的應用軟體之間有頻繁而快速的轉換。

第二類使用者將應用於數據通訊系統，而要求系統具有多工作處理和多使用者處理的能力，以便操作商業應用系統。他們往往在使用 Lotus 1-2-3 或 Wordstar，需要從其他資料庫提取系統(例如最普通的 dBASE II 或 dBASE III等)中取得資訊。他們亦可能需要管理一個小型的會計系統，在這個系統中，可能有兩個至三個人同時使用同樣的一部 PC。

軟體發展商將會成爲第三個消費群，這些發展商可能需要在同一個時間做不同的事情，包括編譯一個程式中，同時在編輯另一個程式，更有可能需要在印字機上列印第三個程式。

系統概觀

C/PC-DOS 的售價約爲三百美元左右，其中包含了一份手册，以及五張磁碟。這些文件可以分爲兩大部份：User Guide；而另一部份稱爲"如何開始使用 Concurrent PC-DOS"。

此外，Digital Research 亦爲 C/PC-DOS 出版了一本程序員指引，其中描述了函數的呼用，以及其他的技術性資料。不過，這份程序員指引是要分開購買的，而非跟其他的文件一起附送。

CONCURRENT PC-DOS

C/PC-DOS 並沒有像 IBM-DOS 那樣包含了應用程式，如 DEBUG，8088 組合器等。相反，這些程式被收錄在一張售價約七十美元的程序員套裝軟件內。這套軟件包含有組合器(Assemble-r)，函數庫管理程式，連結器(Linker)，交叉參攷程式(Cross-Reference Utility)，視窗原程式碼等。文件方面則包括一本使用者手冊，一份 GSX（繪圖功能）用的程序員指引，以及一份 C/PC-DOS 用的程序員指引。

C/PC-DOS 需要使用最少 256KB 的記憶體，以及兩部以上的磁碟機。一種比較理想的硬體組合，是 512KB 以上的記憶體，再加上一部硬磁碟機，及一部軟磁碟機。

與其他系統的相容性

基本上，C/PC-DOS 可以與 IBM-DOS 和 CCP/M-86 相容，但是實行起來却仍有部份的問題存在。

若 C/PC-DOS 被裝置(Install)於硬磁碟機中，我們可以把它放置於 IBM-DOS 之下，或是 CCP/M-86 之下。若果 C/PC-DOS 安裝於 CCP/M-86 之下，就祇有 CCP/M-86 的軟體可以被執行。但若 C/PC-DOS 安裝於 IB-M-DOS 之中，那麽，不管是 CCP/M-86 或 IBM-DOS 的軟體都可以被執行，此時，所有 C/PC-DOS 的有關檔案都會被抄進主目錄(Roo-t-Prefix)中，而現行的 IBM-DOS 副目錄結構(Sub-Prefix)會被保留，以達致相容的效果，於是，C/PC-DOS 就可以讀取並執行 IBM-DOS 及 CCP/M-86 兩者的軟體了。

要將 C/PC-DOS 安裝於 IBM-DOS 之中，必須執行一個特別的設置程式。如果主目錄已經有一個 AUTOEXEC.BAT 檔案，這程式會將它的名稱更改，並產生一個新的 AUTOEXEC.BAT 檔案，這檔案會讀入 CCPM.SYS 檔案，所以每當 IBM-DOS 被啓動時，C/PC-DOS 亦會被自動讀進微電腦中並取得控制權。

CCPM.SYS 是 C/PC-DOS 的基本部份，它是用來處理 CCP/M-86 和 IBM-DOS 程式的系統需求。當一個 IBM-DOS 程式在運行的時候，CCPM.SYS 將所有的作業系統中斷(Inte-rrupt)和函數呼用(Function Call)改成指向駐於記憶體中的 IBM-DOS 軟體副程式。IB-M-DOS 祇向一個程式要求系統服務，這程式就是 CCPM.SYS。當執行一個 IBM-DOS 程式時，CCPM.SYS 使用 IBM-DOS 的檔案格式。而當一個 CCP/M-86 的程式在運行時，CCP-M.SYS 亦提供了完全的 CCP/M-86 相容性。

C/PC-DOS 可以產生檔案，格式化，或被讀進於 CCP/M-86 或 IBM-DOS 的磁碟中，檔案 LOADCCPM 便是用來在硬磁碟機的 IBM-D-OS 和 C/PC-DOS 兩種格式之間作轉換控制，而毋須將檔案重新讀入系統中。

可是 C/PC-DOS 有一個很明顯的限制，就是它跟 IBM-DOS 的相容性祇限於 1.1 版本。因此，大部份 2.0 及以後版本的 IBM-DOS 功能都不能運作。也就是說，使用到 ECHO，SHIF-T，IF，FOR 和 GOTO 命令的批次(Batch)檔案均不能執行，而那些需要 IBM-DOS 2.0或3.0 作為系統的程式亦不能使用 C/PC-DOS。

另一方面，C/PC-DOS 亦提供了一些 IBM-DOS 2.0 的功能，例如硬磁碟機，IBM-DO-S 的索引結構，Backup，恢復(Recover)，MKDIR，CHDIR，RMDIR 和 360KB 的磁碟驅動器格式等。

如果在 IBM-DOS 磁碟中有一個 CONFIG.SYS 檔案，那麽 C/PC-DOS 在通過 CCPM.SYS 取得控制之前，CONFIG.SYS 會先被執行（並設置好各磁碟驅動器）。當然，若這些設置好的磁碟驅動器需要特殊的 IBM-DOS 2.0 系統功能，它們就絕對無法運行。

C/PC-DOS 的另一個主要弱點就是它沒有對應於 IBM-DOS 的 PATH 命令的 PATHNAME 搜尋功能。缺乏了這項功能，處理程式和檔案將會困難得多，要執行不同的應用程式便需要更改副目錄。

為了允許對目前的目錄以外作有限度的存取，C/PC-DOS 提供了兩個浮動驅動器(N：和O：)以及重定義系統碟的能力。藉着使用 CHDIR 和 SYSDISK，浮動驅動器和系統磁碟可以對應於任何次索引和驅動器。若果一個程式並非處於目前的驅動器和次索引中，系統磁碟會被自動搜尋，而一個檔案的位置亦可以藉着在檔案名稱前冠以 N:或O: 以指出其位置。

一些普遍的 IBM-DOS 軟件曾被測試是否可在 C/PC-DOS 上，結果顯示出不相容的問題主要可分成兩大類，顯示輸出的問題，以及 IBM-D-OS 2.0 的函數呼用。

在 C/PC-DOS 之下，是不能使用功能鍵 F1 到 F5 來作為編輯用的。這些鍵因為可以對曾鍵入的命令加以編輯，所以使得鍵盤輸入更加容易，另一方面，利用 C/PC-DOS 中的一個指引導向程式 FUNCTION，可以很輕易的將功能鍵程式化。

CONCURRENT PC-DOS

在目前的索引錄中，兩個副目錄之間的轉換命令，例如 CD C：＼SUB1＼SUB2，對於 C/-PC-DOS 來說，是不合規格的，取而代之必須使用兩個命令，第一個是 CD SUB1，跟着是 C-D SUB2。

另一個比較次要的不相容性是標準裝置名稱，它們在 C/PC-DOS 中和在 IBM-DOS 中的名稱並不相同。例如，串聯介面埠在 IBM-DOS 中稱為 COM1：和 COM2：，而在 C/PC-DOS 之中却分別稱為 Port1，以及 Port2。

BASICA.COM 亦顯示出一種型態上的不相容性。因為 BASICA 會控制所有的 PC 硬體，它並不能於同一時間顯示多於一個的視窗。當然，若這情形發生，使用者立刻就可以知道，可是，系統亦必須再重新起動。

畫面輸出問題亦不會過於嚴重，C/PC-DOS 可以運用 IBM-DOS 的程式，儘管這些程式需要利用 IBM-DOS 的畫面輸出函數調用，或是 IB-M BIOS 的調用，以及對畫面輸出硬件或記憶體作直接的處理。可是，當一個 IBM-DOS 的程式以為它可以將輸出寫在螢幕上任何的地方，那麼四個視窗將會被破壞。如果程式輸出的地方超出了 C/PC-DOS 分配給它的範圍，其它的視窗就會變得十分混亂了。不過，C/PC-DOS 已經將問題儘量減低，它可以允許將視窗重寫，如此各視窗原本的內容就可以輕易地恢復。

視窗邊界的問題是 C/PC-DOS 系統中包含有 SUSPEND 命令的理由之一：它可以允許我們將不處於活躍視窗下的程式操作暫停，以免這些視窗的顯示畫面被破壞。

表一是一個 IBM-DOS 程式在 C/PC-DOS 下的測試結果。注意，大部份基於 IBM-DOS 2.0 或以後版本而發展出來的程式都不能操作，因為 IBM-DOS 2.0 的函數呼用會令得 C/PC-D-OS 完全不能運作。不過，Digital Researc-h 已經防止了這種情形的發生：當程式需要執行任何不合法的函數呼用時，系統會顯示出錯誤訊息，並將程式終止。雖然程式終止，但這比破壞整個系統較為好了一點。由表一可以知道，大部份舊版（1.1）的 IBM-DOS 程式都可以在 C/PC-D-OS 下執行，而 2.0 以後的程式却不能在 C/P-C-DOS 下執行。

表一：在 C/PC-DOS 下所測試過的 IBM-DOS 程式，有星號的程式為可相容的軟件

PROGRAM	COMPATIBILITY
*Mome Accountant Plus	Y
*MultiMate	Y
*MultiPlan	Y
*Lotus 1-2-3	Y
*pfs:graph	Y
*pfs:report	Y
*pfs:write	Y
*SuperCalc 2	Y
*SuperCalc 3	Y
*VisiCalc	Y
*WordStar	Y
IBM Logo 1.0	Y
WordPerfect 3.0	Y
SSI P-Edit	Y
Turbo Pascal 2.0	Y
IBM Macro Assembler 1.0	Y
Crosstalk XVI 3.4	N
Sidekick 1.1	N
Spotlight 1.0	N
PC-Calc 1.06	N
dBASE II 2.4	Y
dBASE III 1.0	N
VDISK.SYS(IBM DOS 3.0 RAM disk)	N

多工處理

對於在 IBM PC 上運行的 CP/M 作業系統家族來說，多工處理（Multi-task）並不是新鮮的事情。Digital Research 所出品的第一個多工處理作業系統，是 CCP/M-86。稍後，帶有視窗的 CCP/M-86 被推出，而現在，則是 C/PC-DOS。

C/PC-DOS 的多工處理系統，目標是讓每一個使用者可以同時處理多至四個不同的程式。這些不同的程式，用 C/PC-DOS 的術語來說，稱為虛擬終端機。

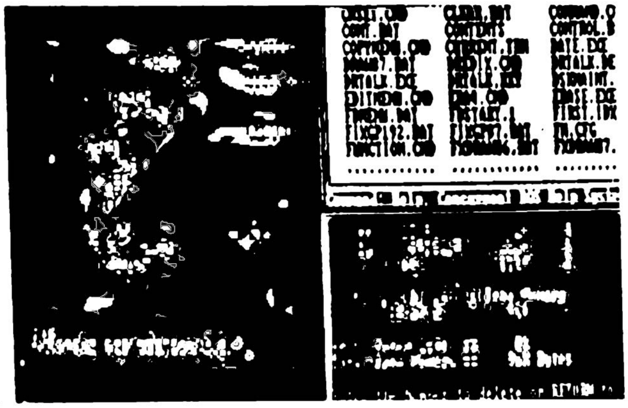
多工作處理系統的好處是可以充份利用計算機的等候時間，例如等候按鍵輸入等。這些等候時間儘管祇需數毫秒，但已是足夠讓其他的程式去運作了。程式以循環式取得控制，當現行的工作需要等候 I/O，或所分配的時間用完，其他的程式就會被觸動。

不同的虛擬終端機之間在 PC 上的顯示可以利用 Ctrl-1 到 4 來控制，這些數目必須在數字鍵盤上輸入。若這些按鍵的定義跟 IBM-DOS 程式發生衝突的話，FUNCTION 程式可用來重新定義觸動虛擬終端機之按鍵，來避免衝突。

當一個虛擬終端機正處於活躍（Active）狀態，它所對應的視窗會於螢幕某處顯示。每一個視窗都可以利用 WMENU 重組以控制它的大小，色彩以及視窗於螢幕上的位置。四個視窗的大小，可以被定義成以複疊的形式顯示，或以緊貼的形式（所有四個視窗全部顯現）來同時顯示。若果視窗之間有所複疊，那麼正在活躍的視窗會全部顯示，而其

他三個視窗則會儘可能顯示。照片一所示爲四個緊貼的視窗的一部份。

照片一：四個緊鄰的視窗



每一個虛擬終端機都有一個啓動批次檔（相似於 IBM-DOS 中的 AUTOEXEC.BAT）以使得當 PC 啓動時，不同的程式可以在每一個視窗中運行。

先前提及的 SUSPEND 命令，可允許或防止目前工作以外的程式運行，是用來加速程式的執行，代價是其他程式的速度減慢了。這命令亦可以用來減少視窗間的混亂狀況。

於作業系統中增加多工作處理功能必須增加一些相關的特性。PC 的資源如檔案、印字機及顯示螢幕等，必須加以保護，以防止被重覆的工作所使用。C/PC-DOS 提供了密碼保護和幾個服務程式：USER，PRINTER，PRINTMGR 等來管理這些資源。

USER 可以替目前的使用者於十六個使用者號碼中選擇一個，C/PC-DOS 的檔案和印字機都可以被定義爲某一特別使用者的個別使用，以防止其他使用者使用。USER 0，亦即系統使用者，具有特別的權力，可以處理系統中的任何檔案和印字機。

PRINTER 用來將一部指定的印字機指派於一個特定的視窗之中。它可以將多至五部印字機指定於同一視窗，或使四個視窗共用一部印字機。

PRINTMGR 用來管理所有視窗同時印出的工作。它處理一個檔案排除序列，以指出每一視窗需要印出的內容，以防止不同的工作造成印製報表的混亂。

C/PC-DOS 提供了密碼保護，SET 命令對每一檔案定義出需要密碼輸入的檔案操作。從完全不受限制到每一操作均需密碼都可。SET 可以使用通用字元（Wildcard），所以一系列檔案可以有同一的密碼。

USER 和密碼保護，是多工作處理系統中，對檔案作保護的兩個工具。但却不能用於 IBM-DOS 環境中。所以必須極之小心，以防止 C/PC-DOS 將這些 IBM-DOS 的檔案破壞。

當多工作處理系統使用時，記憶體變成一項頗爲缺乏的資源。因爲 C/PC-DOS 本身已經使用了差不多 156KB 的記憶體，所以很難再於最低要求的 256KB 中同時執行兩件工作，對於任何的 PC 來說，使用 640KB 的 RAM 比較適宜，雖然這樣並不保證可以同時運行四個中型至大型的程式。

多用戶功能

C/PC-DOS 的多用戶（Multi-user）功能可以讓兩個使用者終端機連接於串聯介面上。終端機可以是 IBM-PC，其他 PC 甚至是無智慧型終端機均可。多使用者的終端機可以直接經由 RS-232 或是經由調制解調器（Modem）連接於主機上。完全運用 C/PC-DOS 的多用戶功能，可以同時處理六件工作。當終端機與主機連結後，終端機的螢幕完全相當於一個完整的 C/PC-DOS 視窗。

要建立一個 C/PC-DOS 多使用者終端機的第一步是安裝程序。要利用調制解調器將 PC 與終端機建立起溝通，需要花費大約一個小時。

於終端機上使用系統有些限制：每一次祇能有一個程序在運行，所以在終端機上並無法控制視窗。要使用功能鍵作輸入的程式也不能執行。要直接控制畫面顯示硬體的程式也不可以在終端機上執行，因爲這些程式會使得終端機被鎖起，而變成無法使用。此外，處於 C/PC-DOS 顯示螢幕底部的狀態欄，用來顯示出目前正處理中的程式，使用中的磁碟機，可用的印字機，以及系統時間等，這些都不會出現在終端機的螢幕之上。

C/PC-DOS 提供了一個命令＜STOP＞，用來終止在 PC 上運行的工作。工作結束後，就可以將記憶體空出以供其他工作之用。

當我們使用 IBM-DOS 操作環境時，每一個使用者並無法防止其他使用者的無意破壞。密碼保護和 USER 命令都不能使用。浮動驅動器和系統磁碟的目前定義，可由全部的使用者來分享，這有可能會帶來一些災難性的後果。例如，假設使用者 A 設定了 N：＝C：＼SCRATCH 並準備將該副目錄中的所有檔案全部刪除。如果此時，使用者 B 設定了 N：＝C：＼以進行他所需的工作。那麽，當使用者A下達 DEL N：＊.＊之命令時，所有

主目錄中的檔案，包括全部的 C/PC-DOS，都會被刪除。所以使用時必須非常小心。

效率的比較

對於任何多使用者系統來說，速度是一個很重要的特性，如果夠快的話，其他的缺點也許可以容忍。可是，非常不幸，C/PC-DOS 的表現並不理想。

當然，C/PC-DOS 的緩慢是有理由的。多使用者和多工作處理功能都需要一定數量的額外程序來管理個別的工作和它們的資源分配。此外，IB-M-DOS 跟 IBM-DOS 程式之間的 C/PC-DO-S 介面亦加重了系統的負荷工作量。

於 C/PC-DOS 中執行 IBM-DOS 的內置指令非常緩慢。例如 CHDIR，原本是 IBM-DO-S 的內置命令，長駐於記憶體中，可快速地執行，但透過 C/PC-DOS，却必須從磁碟中載入，並在一個負荷甚重的系統中運行。

另一個測試是利用 dBASE II 來修改一個紀錄，並對整個資料庫更新索引。這個資料庫包含有 254 個紀錄，每個紀錄有 225 位元組的長度，並分爲十三個不同大小的欄位以及一個鍵值。這一件實際的工作在 IBM-DOS 下運行需時 33 秒；單獨一件工作在 C/PC-DOS 下運行需 45 秒，兩件工作同時在 C/PC-DOS 下運行則需 116 秒。

單獨執行一件工作，C/PC-DOS 比 IBM-DOS 慢了 36%，注意兩件工作同時在 C/PC-DOS 上執行要比順序單獨處理多用 26 秒（116 秒對 45 秒的兩倍），因此多工作處理系統會多用 29% 的時間。兩件工作若順序使用 IBM-DOS 來處理，祇需 66 秒，換言之，同時使用 C/PC-DOS 處理會比順序使用 IBM-DOS 處理慢了76%，而其他的測試，亦可得到大致相同的結果。

一套額外的測試可用來鑑定多使用者操作的表現。三個多使用者的操作會被進行測定，將一檔案印出到 PC 和終端機的螢幕上去；利用 MASM 去組合一檔案；以及在次索引內進行檔案複抄。測試的結果示於圖一。

第一個試驗是印出一個長達 43264 位元組的檔案。終端機以每秒 1200 位元的調制解調器連接。傳輸時間是使終端機印字速度減慢的最主要原因。雖然使用一條每秒 1200 位元的傳輸線去傳送一個這麼大的檔案需時 360 秒，而其他的工作祇需 2 秒，可是檔案的傳送與其他工作是同時進行的。因此有大量的時間被用來等候輸送完成，此即提供了一個機會使 C/PC-DOS 有效地處理多工作處理系統。

圖一：多使用者功能的效率比較

OPERATION	SPEEN IN SECONDS PC	TERMINAL
Type		
Single user	79	362
Multiuser	99	362
Masm		
Single user	45	45
Multiuser	147	147
Cope		
Single user	48	46
Multiuser	95	95

若果使用每秒 9600 位元的直接連結，檔案傳送的速度會快 8 倍，祇需 45 秒，傳送和其他控制的重疊時間減少了，所以若用每秒 9600 位元的傳輸線，TYPE 的速度可以比得上主機的速度。

若果終端機上同時下達了 TYPE 命令，主機的執行速度會變慢（99 秒對 79 秒，慢了 25%）。終端機上執行的速度却不會改變，仍是 362 秒，不論工作是同時抑或是單獨進行。

在終端機上印字不會受主機上的工作減慢的原因是當資料完全傳送到終端機之前，C/PC-DOS 有充份的時間去處理其他工作。可是，在主機上印字，却會受到終端機上的印字工作所阻延，因爲，當一個檔案在 PC 上印出時，並沒有任何的時間阻延，故 C/PC-DOS 一定要將這件工作暫停，以處理其他的工作。

圖一亦示出利用 MASM 來組合一個 10682 位元組（10KB）的檔案需要多長的時間。MASM 於多使用者狀態下使用終端機是非常有效的，終端機上和 PC 上的組合時間完全相同，每秒 1200 位元傳輸線的影響可以減到最少，因爲 MASM 是一個使用 CPU 非常多的程式。不過，這個測試亦顯示出當兩件工作同時進行時，結果非常令人失望（同時進行的 147 秒對連續執行的兩倍 45 秒一一多出了 57 秒）。使用 C/PC-DOS 同時進行組合會比順序組合多花費 63% 的時間。

在 PC 上和在終端機上的 COPY 速度是可以比較的。多工作處理所花的時間差不多可以忽略（雖然多工作處理複抄並不快於分開兩次來複抄），兩個 COPY 程序的多工作處理系統是頗爲有效的，因爲 C/PC-DOS 可以利用大量的 I/O 的好處來處理控制的過程。

另外兩個測試的結果也是值得一提的。首先P

-IP 用來將檔案在不同的次索引間傳送，結果跟 COPY 差不多。其次，在主機上和在終端機上將一檔案印出報表所需的時間相同，因爲終端機祇是用來啓動主機的工作而已。

當很多工作需要執行時，連續執行通常會比同時執行爲快，而當有多使用者時，需大量記憶體的功能可以在終端機上以跟主機一樣快的速度執行。若果一件工作需要有大量的數據傳送，最好不要在終端機上執行，因爲傳送的速度實在太慢了。

結論

C/PC-DOS 本身是相當成功的系統，但根據上文所述，三個主要的弱點使得 C/PC-DOS 的性能大打折扣。

首先是速度方面，多用戶或多工處理的能力事實上並沒有帶來甚麽好處，試驗證明程式順序執行會比同時執行更爲快速。

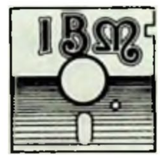
第二個主要的弱點是不能使用 Pathname。Digital Research 原本計劃改良 C/PC-DOS，使得它可以完全通用於 IBM-DOS 2.1，包括 Pathname。而將有更多近期的 IBM-DOS 程式可與 C/PC-DOS 相容。

第三個主要的弱點就是對於 IBM-DOS 的操作環境缺乏保安性。在這些磁碟上的檔案事實上是無法防止被同時存在的其他工作或使用者所毀壞。Digital Research 計劃即將開始 C/PC-DO-S 的第二個改良計劃，這個版本會支持 IBM-D-OS 3.0，更主要的目的是對檔案和資源提供必須的支持。

到目前爲止，對於那些習慣於使用 Concurr-ent PC 特性的使用者來說，C/PC-DOS 3.2 是值得推薦的，但有一個大前題，就是對於它的性能不能立刻要求太高。

對那些需要使用有限度的多使用者功能的人士而言，若果終端機的使用是頻繁的但非長時間連續使用，C/PC-DOS 可能非常有用。最後，對於那些希望在 PC 上同時執行 IBM-DOS 及 CCP/M-86，或是祇使用 CCP/M-86 的使用者來說，它亦是頗有吸引力的。這些使用者將可以使用所有在 CCP/M-86 版本中不能使用的 IBM-DOS 軟件。■

PC-DOS 3.2 有 BUG?



現時最新版本的 PC-DOS 是 3.2，我們已於微電腦技術 V2#6 介紹過各項功能及特點。但是大家在使用 PC-DOS 3.2 之時，有沒有遇到下列情形？

BOOT 起了 PC-DOS 3.2 後，放進另一張磁碟（非 PC-DOS 3.2 格式）執行一些檔案之時出現"FILE ALLOCATION TABLE DAMA-GED"之訊息，無論 A、B 磁碟機也是一樣，而用 CHKDISK 又沒有任何問題。

究其原因是 PC-DOS 3.2 執行時會讀取生產商的名字，以及 BIOS 參數區的磁碟格式，這些資料可以在啓動磁區（TRACK 0，SECTOR 0）找到。MICROSOFT 及 IBM 皆指定生產商名字出現在啓動磁區的第 8 BYTE，一般情形之下這個名字是"I-B-M"連同 DOS 版本號碼。較早期的 DOS 版本會忽略這些資料，而 PC-DOS 3.2 則堅持需要這些資料，否則便不會讀取這些磁碟。

由於 IBM 售賣 PC-DOS 之時要向 MICR-OSOFT 註冊（PC-DOS 是 PC 版本的 MS-DOS），自然 IBM 不希望支持其他非 IBM 的 D-OS。要解決這個問題，是有以下兩個方法：

(1)首先 BOOT 起較早期的 PC-DOS，然後把檔案抄錄至剛以 PC-DOS 3.2 FORMAT 好的空碟。

(2)又或者可以利用 DEBUG，把 PC-DOS 3.2 需要的資料抄回。

a. 從 DOS 載入 DEBUG
   DEBUG

b. 放進 PC-DOS 3.2 能讀取的磁碟（非啓動磁碟）
   L100 00 1

c. 放入有問題出現的磁碟
   W100 0 0 1

d. 離開 DEBUG
   Q

■

多用戶資料庫系統：PROGRESS

■ 李兆基 ■

一間較具規模的公司往往需要一個多用戶（MULTIUSER)系統，而此系統必需具有多重視窗畫面格式，有處理複雜的聯系性資料庫結構的能力及可算完美的結構保護，同時具有高階語言供用戶自行編寫應用程式。由DATA LANGUAGE CORP 發展成功的多用戶系統PROGRESS(簡稱PGS)包括以上特點，對於資料庫程式編寫員來說，這系統確給他們帶來更多便利，筆者在此極力向大家推薦。PGS的規格詳列於附表一。

PGS是一個完備的聯系性資料庫管理系統，具有高階結構程序化語言，卓越的多用戶設備，供資料庫元素的欄層面保護，同時有最佳的支援。各工具通用於MS—DOS(單用戶)及UNIX和XENIX的多種版本，因此可以在普及的IBM PC發展龐大的系統，之後輕易載入UNIX操作環境。

提供的文件

PGS包括自學及參考兩本指南，一本指令摘要和一塊功能字鍵模板。對初學程式編寫的朋友來說，自學指南幫助不少。參考手册不是用來學習PGS，而是詳列出特殊要求，功能等重要資料，這兩册都加上了充足的附圖。

ON—LINE HELP極之出色，它包括提議對錯誤信息如何反應。這份文件只提供唯一版本，原因是這語言在任何操作系統都是一致的。當然，不同的裝置手册會適用於不同的操作環境。

資料字典

如同大部份資料庫系統一樣，PGS亦需要閣下首先描述閣下的資料庫，一個以目錄操作的字典系統爲龐大的資料庫應用定義充足數目的檔案，欄及檔案索引。資料庫內可允許總數達1023個索引，而每個索引最多可以有10個欄。因爲每個檔案最少一定要有一個索引，固此檔案極限數目爲1023個。而任何數目的索引可以爲任何檔案被定義。

PGS支持字符、整數、點數、邏輯(眞/假)，及日期欄同時對日期作出正確運算(例如，兩個日期之間的差是一個數字等)。啓動選擇允許挑選歐洲日期及點數格式，好似86-12-30及12，34。筆者對使用格式的彈性十分欣賞，它包括供邏輯數值顯示的巧妙方法。下面的定義：

```
Define variable SAFETY-CAP as logical
                format "Safety/Non-safety"
```

結果將放入一變數內，當眞時顯示"Safety"，而假時顯示"Non-safety"。若要更新，只要輸入首字母"S"或"N"便會轉變整個數値之顯示。

所有欄會被貯存在可變長度格式內，因此整個資料庫的體積可以比傳統系統減低一半，而且免除定義額外的"Extension"檔案來存放額外的長欄。

資料核對片語(及相關的錯誤信息)可以就任何欄而被輸入，而且能夠參考任何其他資料元素。舉例說，證實片語爲一欄ITEM-ORDERED或會是"Can-find(ITEM where ITEM-CODE=ITEM-ORDERED)"

記錄可以全部或部份存取，一個郵寄列表，只需輸入姓氏第一個字母及(可選擇性)證明該人的部份密碼就可以找出記錄。下列句子可尋出某人的記錄或創造一個新的：

```
Find PERSON where LAST-NAME begins NAME-ENTERED
         and ZIP-CODE begins ZIP-ENTERED no-error.
         /* No-error stops the error message
            "Person ... not found"*/

If not available PERSON then /* Didn't find unique record */

If ambiguous PERSON then to
   ... /* Display records and let user choose */
   End.

Else do:                          /* Didn't find any */
   ... /* Create new record */
   End.
```

任何時間都可以加入另外的欄、索引，及檔案，就算當資料庫有活躍的記錄欄內的字元都可以被改變。例如想將一個字元欄取代錯誤定義的數値欄，最簡單方法是加入一個新的欄，轉換及抄去現存的資料，移去數値欄，並將字元欄改名爲原先的欄名。

PGS的資料字典本身也是用PGS編寫，因爲所有檔案被一個 FILE檔案內的記錄所替代，同時欄被通往一個 FIELD檔案內，因此可以能夠編寫閣下特有的資料報告或更新副程式。例如：

```
For each _FIELD where _FIELD-NAME begins "MED":
    FIND _FILE of _FIELD.
    Display _FIELD-NAME _FILE-NAME _FORMAT.
    END.
```

以上的技巧普遍上較爲實用，當有需要改變資料元素時可以有規律地編寫特有的程序(例如將全部欄由X-NUMBER形式改名爲X-NUM)。

處理資料庫的關係

上述例子說明PGS能夠處理檔案之間的關係二個方法。因爲任何欄都能夠被索引，所以通常用存於另一個檔案的數

値在一個檔案內尋出記錄是可行的。因此：

```
FIND ITEM where ITEM-CODE = ITEM-ORDERED
```

會找出ITEM記錄，它的ITEM CODE欄與ORDER-LINE記錄內的ITEM-ORDERED數值一樣，倘若ITEM在ITEM-CODE有索引的話將更有效率。

在不同的檔案內使用相同的名稱，可以更爲稍化：例如欄名ITEM-CODE是用來在ITEM 檔案內定爲項目，並且在ORDER-LINE檔案內亦定爲購買項目，當兩欄被索引(快速存取ITEM記錄，或者對所有ORDER-LINE的項目)，此時可說"FIND ITEM of ORDER-LINE"。可以如此簡化：

```
Find _FILE of _FIELD.

For each ORDER-LINE of ORDER:

For each ORDER of CUSTOMER,each ORDER-LINE of ORDER:

For each COMPOSTION of PRESCRIPTION,
        MEDICINE of COMPOSITION:
```

(注意：當可能有幾個記錄相符時才需列明"each")

PGS其餘句子用作任何已定義之索引或以"preselect"片語之排列定義存取FIRST，LAST，NEXT，或PREVious記錄。

這些句子的強勁功能可以下面一段程式說明，用來修改現存的MEDICINE記錄，倘若需要可創造新的MEDICINE記錄及允許改變爲KEY數値(MED-CODE)：

```
/* CHG-MED Create or change MEDICINE records */

Define variable OLD-MED-CODE like medicine.MED-CODE.

Repeat:

   Prompt-for medicine.MED-CODE.
   Find MEDICINE using MED-CODE no-error. /* no-error lets us
                                             choose to creats */

If not available MEDICINE then do:
   Message " This medicine will be added -- unless you push ESC".
   CREATE MEDICINE.
   Assign MED-CODE.
   End.

OLD-MED-CODE = MED-CODE. /* in case MED-CODE is changed below */

Next-prompt MED-NAME. /* The cursor will start at this field */

Update medicine.MED-CODE MED-NAME MED-FORM PRICE-CODE

If medicine.MED-CODE entered then do: /* Changing MED-CODE requires
                                         Changing other records
                                         with OLD-MED-CODE value */

     For each PRESCRIPTION where presciption.MED-CODE = OLD-MED-
CODE:
        presciption.MED-CODE = medicine.MED-CODE.
        End.
     For each COMPOSITION where composition.MED-CODE = OLD-MED-CODE:
        composition.MED-CODE = medicine.MED-CODE.
        End.
     End.

  End. /* of main loop */
```

適用於PGS的資料庫設計

建設任何成功系統的關鍵是一個色好的資料庫設計，它在正常的情形下能容納所有有關的資料，並允許有效的更新及補救。因爲PGS允許閣下定義一實質上不受限制地在檔案間唯一或非唯一關係的索引，它能夠容納差不多任何資料庫設計。不過錯誤的資料庫設計則會如同其他任何語言一樣在PGS內造成破壞。

用戶友善介面

PGS設計者提供了一完善的設備定義用戶求助功能，PGS程式員可以自行定義他們自己的求助形式。無論何時用家按動HELP鍵(通常是F2)，用戶編寫的爲APPLHELP 副程式便執行。APPLHELP能夠暫時性地以任何文字遮掩螢幕，並且能夠使用所有PGS特色在資料庫任何部份進行詢問或更新。FRAME-FIELD功能決定當HELP被按下時游標指向那一欄，因此求助程式能夠指明這欄。

下面的求助程式是一藥方追踪系統，當游標停在MED-CODE時讓用家驗明藥劑，當停在MD-CODE時驗明醫生。當閣下不想接受PGS的預設値時，它亦說明運用FORM句子指明的螢幕畫面佈局。

```
/* APPLHELP.P  User help for prescription entry */

Define variable MED-START like MED-NAME.
Define variable MD-START  as character format  x(36)" label "MD-name".

                                  /* Help with MED-CODEs */
If frame-field = "MED-CODE" then do with frame find-med.

   Form                           /* Lay out a fancy box on screen */
      medicine.MED-CODE MED-START
      with title "Help -- Identify MED-CODE'
           17 down row 2 frame find-med.

   Set MED-START.                /* User enters part of medicine name */

   For each MEDICINE where MED-NAME begins MED-START:
      Display medicine.MED-CODE MED-NAME @ MED-START.
      Down                       /* Full medicine name displayed there */
      End.

   End./* of medicine name help */
                                /* Help for md codes */
Else if frame-field = "MD-CODE" then do with frame fine-md:

   Form                         /* Another fancy box */
      MD-CODE MD-START
      with title "Help -- Identify MD-CODE'
        17 down row 2 frame find-md.
Set MD-START                 /* User enters start of MD last name */  /

For each DOCTOR where MD-LASTNAME begins MD-START:
   Display MD-CODE string (MD-LASTNAME) + ". " +
                           MD-FIRSTNAME @ MD-START.
   Down.                  /* Display is in form:  Fink, Arthur  */
   End.

End. /* of doctor name help */

Else do:                      /* Cannot ask for other kinds of help */
   Message "Sorry -- no help available".
   End. /* of no other help available */
```

當呼叫求助功能時，APPLHELP亦同時提供一更爲常用的工具允許依次序中斷一PGS程式而執行另一程式。

PGS資料字典備有任何欄的求助功能，無論何時游標停在該欄，都允許本身指示資料輸入。

PSG程式編寫環境

儘管程式可以用其他文字處理器來編寫，內置的全螢幕編輯器却較爲有效，程序可以交談式測試，顯示清楚的錯誤信息。On-line求助提供所有錯誤信息的詳細解釋，且有鍵盤功能參考等。

雖然筆者希望功能包括搜索及替代，游標以每字移動等特色，不過編寫環境已使人滿意。編輯器自動地在縮入的行號下縮進去，令鍵入結構化程式更容易。

發展商如不希望保留原始碼在他們的產品，便可使用RUN-TIME版本的PGS，它只可以執行這些預先編譯的程序。一個類似的Query-run-time產品只會執行這些程序及顯示要求。

多用戶資料庫系統：PROGRESS

畫面及報告格式

PGS提供優良的格式化複雜畫面，包括可重覆地隱蔽重現的多重視窗。畫面分爲多個Frames，它可用作提示字元及顯示一組要求的句子。Form 指令能夠用來明確地分佈Frames，或者Frames能夠自動地被設定及顯示於螢幕。Frames 的概念對大部份程式員來說非常新穎，初次使用或會有點混淆，學會後就像正常的處理方法無異。

記錄與資料庫之一致性

PGS內的基本單位是記錄(TRANSACTION)，對資料作一系列改變時只可以是確定或取消(資料庫無影响)。這些記錄是資料庫完整的要素，允許強力的用家資料確定及控制多用戶操作功能。

考慮下面的程序，它由CUSTOMER檔案內的BALANCE-DUE 減去一顧客的支付款項，並爲AMOUNT-PAID及CUSTOMER-ID以數值創造一個ACCT-HISTORY 記錄。

```
Repeat:
   Prompt-for customer.CUSTOMER-ID AMOUNT-PAID.
   Find CUSTOMER using CUSTOMER-ID.
   Create ACCT-HISTORY.                    Assign AMOUNT-PAID
   Acct-history.CUSTOMER-ID = customer.CUSTOMER-ID.
   BALACNCE-DUE = BALANCE-DUE - AMOUNT-PAID.
   End.
```

每個支付款項過程都會產生一個記錄，倘若閣下的系統中途毀壞(或者是在創造ACCT-HISTORY記錄後但在更新CUSTOMER記錄前)，當閣下重新啓動PGS，資料庫會自動恢復爲毀壞前之狀況。保證與預先的資料庫一致。

UNDO指令讓閣下的程式取消記錄成爲部份執行邏輯，並隨之逸往下一資料行，在錯誤行號修改輸入，或完全離開程序。下面的"Undo，retry"句子允許用家固定輸入數值(或按"escape"鍵跳往下一項支付。)

```
Repeat:
   Prompt-for customer.CUSTOMER-ID AMOUNT-PAID.
   Find CUSTOMER using COSTOMER-ID.
   Create ACCT-HISTORY.
   Assign AMOUNT-PAID.
   If AMOUNT-PAID <= 0 then do:
      Message "Payment must be positive".
      Undo, retry.
      End.
   Acct-history.CUSTOMER-ID = customer.CUSTOMER-ID
   BALANCE-DUE = BALANCE-DUE - AMOUNT-PAID.
   End.
```

多用戶操作

相似的記錄進行/取消方案保證資料庫完善亦是PGS支持多用戶的要素，資料更新過程是一連串的記錄，結果不會被其他使用者看到直至全部完成爲止(因此不能取消)。

通常情形下，PGS指明記錄與細小工作個體(對多用戶更新非常有效)，不過可指定較大或較小的句子組合被考慮爲記錄。舉例說，一連串的借貸若果平衡則以一單元接受，或全部不接受。預先定義一記錄區環繞所有借貸過程，閣下可保證直至整區完成前任何用戶均不會看到這些改動。這情況下，PGS會自動鎖起已被更新之記錄，當記錄完成後才加以釋放。

筆者發覺這系統容易明白及操作，絕少需要顧慮鎖上記錄與否，不過要考慮到更新多重檔案時記錄的範圍（正如上文提到的借／貸例子）。所有多用戶操作的指令都允許在單用戶的PGS內，因而多用戶系統能夠在MS-DOS環境下被發展。

軟件的錯誤

不斷測試後，筆者在制作版本的PGS找到幾個BUGS，它們都是難以置信的失色。

其中一樣是PGS並不傳統，同時要慢慢熟習。對於簡單的報告Frames的畫面格式提供出色的預設畫面，但是精確的多重視窗畫面顯示定要小心處理，要多實習才能徹底明白Frames如何控制畫面。

筆者希望PGS文件提供的副程式更具彈性，更佳的文字操作設備(雖然PGS內的功能允許編寫有需要的常規)，及多幾個除錯工具。然而PGS是一強勁，文件齊備及足以未來發展之強大程式的支援，是目前供MS-DOS的最強勁的資料庫處理系統。

操作系統的考慮

PGS至少需要512K(最好是640K的PC/XT)並最少要有2.5megabytes的硬磁碟機空間供程式存放。PGS運行中可執行其他MS-DOS程式，同時由其它資料庫系統輸入資料。

爲著執行多用戶PGS，筆者運行XENIN於AT 兼容機上。儘管在這新環境下載入PGS應用沒有麻煩，筆者發覺第一次設定整個XENIX系統是一件相當大的工作。最終能夠容納4個或5個用戶，連同4部印字機。倘若日後需要較大的Unix環境也不需把PGS轉換。

在Xenix環境完成系統修改後，筆者將一示範程式轉至MS-DOS系統，並不需任何改動下運行。PGS的文件提供包括一段基本DOS-Unix的差異，正如PC的25行螢幕不能放入24行的VT-100，大小寫程序名稱在Unix是有分別的，但MS-DOS則沒有等。

雖然供MS-DOS使用的PGS版本定價爲695美元（多用戶Xenix及Unix版本按機件容量而定價1450美元以上），却要加上MS-DOS用戶每年$200美元保養費，才能得到所有產品的UPDATE版本。用戶如選擇不簽署這保養計劃則可在日後付出額外費用得到保養。

PROGRESS(VERSION 3)可以由Data Language Corp，47 Manning Road，Billerica，Mass. 01821 購買。■

商用軟件磁碟保護技巧淺談

■ 張峰寧



前言

目前市面上討論 APPLE II磁片之破解及拷貝(COPY)的書籍十分多，但有關 IBM 的應用技術却不多見。在此筆者不自量力地提出拷貝利用 PROLOK 及 SOFTGUARD 保護的磁碟之方法；望此輩高人多多指教！

拷貝 PROLOK 保護之磁碟

一些經過 PROLOK 保護處理的磁片會在磁片某些區域留下雷射孔(即由雷射所造成的損壞痕跡)。用 PROLOK 保護的磁片由於這些雷射孔的關係，所以在讀資料及寫資料時會有所差異；而普通程式拷貝出來的磁片由於沒有上述特性，所以便不能工作。

首先利用 COPYWRITER 拷貝用 PROLOK 保護的磁片；在拷貝完後，COPYWRITE 會出現一訊息提醒你在使用拷貝出來的磁片之前，必須先執行在 COPYWRITE 磁片上的一個程式 —— RA-MKEY。只要 RAMKEY 程式被存在 MEMORY 中，你便可以成功地執行拷貝出來的磁片。但請留意若你事先沒有載入 RAMKEY 便嘗試執行拷貝出來的磁片，可能會損壞磁片，而要再次重新 CO-PY！

SOFTGUARD 磁碟保護

第一種方法是要利用套裝軟件 DISK MECH-ANIC V5、11B 中的 UNGUARD.COM 檔案；爲了使各位更加明白，筆者在此擧一例以說明之：——(COPY dBASE III V1.1)

1. 將 UNGUARD.COM 抄至一張 BLANK DIS-K；
2. 將用 SOFTGUARD 保護的原磁片，即 dBAS-E III 放入 DRIVE A；存有 UNGUARD 的 DISK 放入 DRIVE B；
3. 鍵入：
   b:unguard dbase.com b:dbase.exe
4. 之後 UNGUARD 便會破解 dbase.com 的檔案，並在 DRIVE B 的 DISK 中創造出一個沒有保護的檔案；
5. 重新啓動 DOS 並鍵入：
   erase b:unguard.com
   copy a:*.* b:
   erase b:dbase.com
6. 大功告成！之後你這張 DRIVE B 的 DISK 便是一張無保護的 dBASE III 了！

至於其他程式只須照上述方法 COPY 即可！(只須將 dBASE 改爲其他名稱)

而第二種方法是利用 COPY II PC (V2.5 或以上)將 DISK 抄下；之後再執行 COPY II PC 中的 NOGUARD.COM 檔，以後只須根據 S-CREEN 指示去做便可！而 COPY 出來的 DI-SK 便是沒有保護的。

據筆者所知，用 SOFTGUARD 保護的程式計有：

dBASE III V1.1	dUTIL III
SYMPHONY V1.1	dGRAPH III
FRAMEWORK V1.1	QUICKCODE III
FRAMEWORK II	

後記

最後希望這篇文章能有抛磚引玉之作用，望各位讀者能將個人破解及拷貝經驗公開，以公諸同好！下次再見。■

IBM趣味遊戲選

蘋　果　棋

賴耀華

陰陽棋，Reversi，在十九世紀末期才出現的一種戰略棋，曾經在香港流行一段時間，一般稱它爲蘋果棋。這種棋在蘋果電腦上的版本甚多，却少見於 IBM PC 上，於是筆者用 Turbo Pascal 編寫這個遊戲。

首先，特爲初玩者介紹蘋果棋的玩法：蘋果棋的棋子有 64 隻，每隻都是相同而且各有兩面顏色，下棋雙方各佔一色。棋盤由 8×8 格組成，下棋前先在中央四格放下雙方顏色各兩子。下棋者輪次各下一子，而每下一子都必須吃下對方最少一子。所謂吃子，是己方放一子在對方一子隔鄰，而在同一直綫對方棋子的另一端已有己方一子，即對方棋子被己方棋子夾着，則被夾的對方棋子便給吃掉，反轉另一面變成己方棋子。只要頭尾兩端被夾着，則中間所有對方棋子便給吃掉，而且己方放下的一子可對斜綫、直綫、横綫的對方棋子發生效力。當無子可吃時，下棋者便喪失一次下子機會，即讓對方連下兩子。當雙方都無子可吃時，或棋盤放滿棋子，棋局便結束，而盤上棋子較多的一方爲勝者，反之爲敗者。若初玩者仍摸不着頭腦，可試觀看遊戲示範 Demo。

硬件要求

IBM PC/XT/AT 加上彩色圖像卡

軟件要求

Turbo Pascal 3.0

操作方法

程式開始時，電腦要求輸入遊戲狀態 (Mode of Game)，0 代表示範，即電腦對電腦；1 代表人對電腦；2 代表雙人對玩。

- 若選擇 1，按 Y 行先，N 行後。
- 若選擇 2，須輸入先行者和後行者姓名。
- 控制棋子位置可用游標控制鍵。
- 當位置選完後，按 Enter 便可放子。
- 當無子可吃時，按 P 鍵。
- 若按 C 鍵，則電腦代爲下子。
- 若按 B 鍵，可作回棋。
- 在遊戲中途按 ESC 鍵可結束遊戲。

當遊戲完畢，按 Y 鍵繼續遊戲，按 N 鍵表示不再繼續，程式隨之結束。

本程式的重點在於電腦下棋部份，Procedure ComputerMove。雖然本程式不可稱上人工智能，但它的下棋戰略是經過筆者一番研究後才達成的，就算是此道高手也未必可以戰勝電腦。話雖如此，電腦也並非牢不可破。

(Reversi.Pas 爲 Source Program，Reversi.Com 爲編譯後可直接執行 File。)

即期微電腦技術程式磁碟（IBM 版本）收錄了相同名字之檔案。■

新產品介紹

最新擴展咭將PC變爲圖文傳眞機

由 DATACOPY 公司所推出 MICROFAX 擴展咭（1195 美元）能夠使 PC 擁有專業圖文傳眞機（FACSIMILE）同等能力。用戶可以在運行不相關應用軟件之時，同時傳送及接收 FAX 格式的文件。可傳送的資料包括任何 ASCII 檔、任何軟件之顯示畫面、磁碟上的圖像檔案，以及通過接駁影像掃描器（SCANNER）傳送正本文件。

傳送之時不單可以使用 GROUP III FACSIMILE 標準，而且更有一些高級圖文傳眞機才擁有的功能，包括自動撥號、通訊紀錄等。咭上使用的 9600BPS FAX MODEM 傳送一頁資料約需時三十秒，另外還附同第二部 1200BPS HAYES 兼容 MODEM。■

產品名稱：DATACOPY MICROFAX
硬件要求：640K RAM
生 產 商：DATACOPY CORP.
1215 TERRA BELLA AVE.,
MOUNTAIN VIEW, CA 94043

動態記憶管理技巧

大家於寫作低層次的程式中（涉及硬體的輸出／輸入），會否覺得很多時程式都需要大量記憶作爲輸出／輸入的緩衝區？通常的解決辦法只有使用組合器（Assembler）的預留指令——DB，例如下列的程式——READDIR1.ASM（列表一），它讀入磁碟機 A：內的磁碟目錄，並將它存入名爲D-IRECTORY BUFFER 之後的記憶內。以 IBM PC 標準每條磁軌 9 磁區的雙面格式磁碟（360K 格式）而言，那麼緩衝區便需要 7（磁區）＊512（BYTES）＝ 3584 bytes。雖然上述程式只使用了 11 個指令，但當大家將程式組合及用 Linker 將程式連接，再用 EXE2BIN 將 EXE 檔轉爲 C-OM 檔時後，便會發現這 COM 檔案總長度是 3606bytes！那裏來的三千餘 bytes 呢？千餘其實是因爲下列的指令：

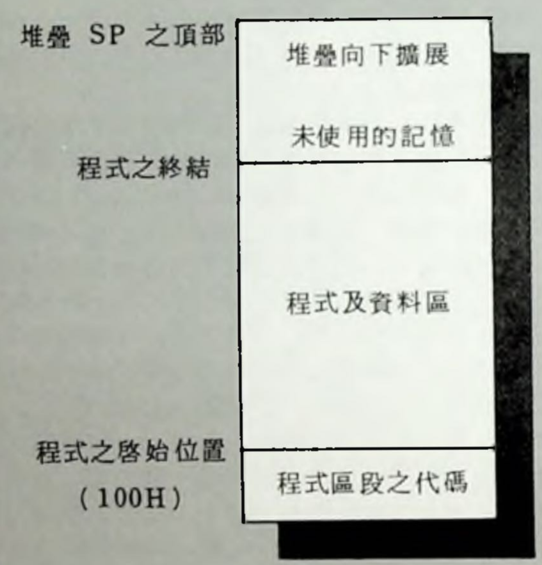
DIR
DIRECTORY BUFFER DB 3584 DUP(?)

使到 3584 byte 的 NULL 被加於 COM 檔案內；這額外的 3 千餘 byte —— 不但浪費了磁碟容量，亦使到程式載入的時間加長。既然上述處理緩衝記憶如此沒有效率，可有其它更佳的解決辦法呢？

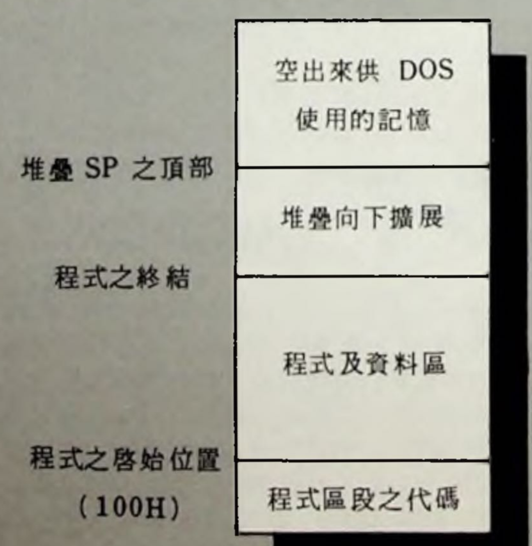
DOS 的記憶管理

每次 DOS 載入一個程式，它都會將全部的可用記憶分配給這程式，但留意是並不包括資料的。因此如要配置一段記憶作爲資料暫存用途，首先便要使用 DOS 的功能 4AH，以減少配置給程式的記憶，再使用功能 48H 要求 DOS 預留部份記憶以作資料的暫存。READDIR2.ASM（列表二）的功能與程式一完全相同，但因依照上述的方法作出修改，所以最後的 COM 檔案只得 58 byte 長，節省了 3.5K byte 的記憶及磁碟空間。因爲 DOS 把 COM 程式都固定爲 64K（圖一），所以列表二之程式首先要求減少 DOS 分配給它的記憶至 64K，再將程式區段地址（program seg-ment address）放入寄存器 ES 內，而要求減少的記憶量存入 BX。最後呼叫 DOS 的功能 4

圖一：程式使用記憶區分佈



圖二：重置堆疊後的 .COM 檔案



-AH 以減少分配給該程式的記憶，當 DOS 將多餘的記憶騰出後，我們便可再將要求的記憶量存入 BX 內，以 DOS 的功能 48H 以將部份記憶保留作爲資料的暫存。上述的方法是可進一步改良的，例如列表二中程式加上資料亦只不過大約 4K，如我們將 64K 記憶分配給它，那實在太浪費 PC的記憶了；對於普通的單用戶操作系統似乎不重要，但對於記憶量需求很大的多用戶、多工操作系統（如 Window，Top-view 等），浪費了的 60K 記憶（相對於 PC 的 640K）便顯得相當重要了！

進一步的改良

列表二之程式內已加入一句指令 INT 03H，使大家在使用 DEBUG來運行程式二時能停止及顯示使用到寄存器的內容，以印証剛才提及的理論。大概讀者亦會發現 CS、DS 及 SS（ES 被程式改變了它的內容）都是相同的數字，代表程式所在的區段，而因爲 DOS 將控制權傳給程式前將 0000 存入 SP，所以現在的 SP＝FFFEH。更詳細的記憶分佈資料可參考圖一，正因爲 Stack 被設置於 64K 的最高記憶，所以每一個程式（CO-M 或 EXE）都最少要佔用 64 記憶（註一），但如我們適當地重新設訂 SP 的數值（見圖二），便可大大減少程式用的記憶，亦即增加其它程式的可用記憶，而列表三便是依據上述的方法而從列表二再改良而成的程式。

註一：除了一些程式會自動再分配記憶或 PC 內的可用記憶少於 64K。■

列表一：READDIR1.ASM

```
;LISTING 1   READDIR1.ASM
;
; This program reads the directory from the diskette in drive A
; into the memory area at DIRECTORY_BUFFER.  For the sake of
; simplicity no console input or output functions are performed
; by this program.
;
CODESEG SEGMENT PARA   PUBLIC   'CODE'
;
        ASSUME  CS:CODESEG,DS:CODESEG,ES:CODESEG,SS:CODESEG
;
        ORG     0100H
;
BEGIN:  JMP     READ_DIRECTORY
;
; ************************ Data Definitions ********************
;
DIRECTORY_BUFFER        DB        3584 DUP (?)
;
; ************************  Main Procedure  ********************
;
READ_DIRECTORY:

  Setup the registers for absolute disk read

        MOV     AL,00H                  ; Drive A
        MOV     BX,OFFSET DIRECTORY_BUFFER   ; Point to the beginning
                                             ;    of the memory area
        MOV     CX,0007H                ; 7 Sectors to be read
        MOV     DX,0005H                ; Start at sector 5
        INT     25H                     ; Absolute Disk Read Interrupt
        JC      ERROR_EXIT              ; Quit if there was an error
        POPF                            ; Restore the user flags
        RET                             ; Return to DOS
;
; Error return point -- error fixup or messages can go here
;
ERROR_EXIT:
        POP     DI                      ; Throw away the user flags
        RET                             ; Return to DOS
;
CODESEG ENDS
;
; ******************************************************************
;
        END     BEGIN ■
```

列表二：READDIR2.ASM

```
;LISTING 2:  READDIR2.ASM
;
; This program reads the directory from the diskette in drive A.
; The input buffer area is outside the program's segment and is
; obtained from the DOS memory pool by calling DOS Function 48H,
; after calling DOS Function 4AH to "shrink" the memory allocated
; to the program.  For the sake of simplicity no console input or
; output functions are performed by this program.
;
CODESEG SEGMENT PARA   PUBLIC   'CODE'
;
;
        ASSUME  CS:CODESEG,DS:CODESEG,ES:CODESEG,SS:CODESEG
;
        ORG     0100H
;
BEGIN:  JMP     READ_DIRECTORY
;
; ************************  Main Procedure  ************************
;
READ_DIRECTORY:
;
; "Shrink" the program memory allocation by calling DOS Function 4AH
;
        MOV     BX,4096         ; 64K for the program
        MOV     AH,4AH          ; DOS function
        INT     21H             ; Function request
        JC      ERROR_EXIT_1    ; Quit if there was an error
;
; Request DOS to allocate a block of memory for the disk input area
;
        MOV     BX,224          ; Room for 7 sectors
        MOV     AH,48H          ; DOS function
        INT     21H             ; Function Request
        JC      ERROR_EXIT_1    ; Quit if there was an error
        MOV     ES,AX           ; Move segment from AX to ES
;
; Set breakpoint for DEBUG so that the registers can be examined
;
        INT     03H             ; DEBUG breakpoint
;
; Setup DS for the Absolute Disk Read
;
        PUSH    DS              ; Save DS
        MOV     AX,ES           ; MOVE the contents of ES to DS
        MOV     DS,AX           ;    via AX

; Setup the registers for the Absolute Disk Read
;
        MOV     AL,00H          ; Drive A
        XOR     BX,BX           ; Point to the beginning
                                ;    of the memory area
        MOV     CX,0007H        ; 7 sectors to be read
        MOV     DX,0005H        ; Start at sector 5
        INT     25H             ; Absolute Disk Read Interrupt
        JC      ERROR_EXIT_2    ; Quit if there was an error
        POPF                    ; Restore the user flags
        POP     DS              ; Restore DS
;
; Free the allocated memory block prior to returning to DOS
;
        MOV     AH,49H          ; DOS function
        INT     21H             ; Function Request
        RET                     ; Return to DOS
;
; Error return without popping the stack
;
ERROR_EXIT_1:
        RET                     ;Return to DOS
;
; Error return with popping the stack
;
ERROR_EXIT_2:
        POP     DI              ; Throw away the user flags
        POP     DS              ; Restore DS
;
; Free the allocated memory block prior to returning to DOS
;
        MOV     AH,49H          ; DOS function
        INT     21H             ; Function Request
        RET                     ; Return to DOS
;
CODESEG ENDS
;
; ***************************************************************
;
        END     BEGIN ■
```

列表三：READDIR3.ASM

```
;LISTING 3:  READDIR3.ASM
;
; This program reads the directory from the diskette in drive A.
; The top of the program's stack is relocated downward in memory.
; The memory allocation for the program is reduced to a size just
; large enough to accomodate the program and its stack.  The
; memory area for the diskette input buffer is located just above
; the top of the stack. The total memory required by the program
; buffer is less than the 64K memory required by a .COM program.
;
CODESEG SEGMENT PARA   PUBLIC   'CODE'
;
;
        ASSUME  CS:CODESEG,DS:CODESEG,ES:CODESEG,SS:CODESEG
;
        ORG     0100H
;
BEGIN:  JMP     READ_DIRECTORY
;
; ************************  Main Procedure  ************************
;
READ_DIRECTORY:
;
; Relocate the stack downward in memory
;
        MOV     AX,OFFSET END_OF_PROG  ; Pointer to the program end
        ADD     AX,256              ; Allow 128 words for stack space
        MOV     SP,AX               ; Move new stack pointer offset to SP
        XOR     AX,AX               ; Push a word of zeroes onto the stack
        PUSH    AX
;
; SP now contains the program length minus 2
;
        MOV     BX,SP               ; Move SP to BX
        ADD     BX,02H              ; Add 2
        MOV     CL,04H              ; Divide by 16 to convert from
        SHR     BX,CL               ;   bytes to paragraphs
        INC     BX                  ; Add one paragraph for safety
;
; "Shrink" the program memory allocation by calling DOS Function 4AH
;
; BX contains the required amount of memory in paragraphs
;
        MOV     AH,4AH              ; DOS Function
        INT     21H                 ; Function Request
        JC      ERROR_EXIT_1        ; Quit if there was an error
;
; Request DOS to allocate a block of memory for the disk input area
;
        MOV     BX,224              ; Room for 7 Sectors
        MOV     AH,48H              ; DOS Function
        INT     21H                 ; Function Request
        JC      ERROR_EXIT_1        ; Quit if there was an error
        MOV     ES,AX               ; Move segment from AX to ES
;
; Set breakpoint for DEBUG so that the registers can be examined
;
        INT     03H                 ; DEBUG breakpoint
;
. Setup DS for the Absolute Disk Read
;
        PUSH    DS                  ; Save DS
        MOV     AX,ES               ; Move the contents of ES to DS
        MOV     DS,AX               ;    via AX
;
. Setup the registers for the Absolute Disk Read
.
        MOV     AL,00H              ; Drive A
        XOR     BX,BX               ; Point to the beginning
                                    ;    of the memory area
        MOV     CX,0007H            ; 7 sectors to be read
        MOV     DX,0005H            ; Start at sector 5
        INT     25H                 ; Absolute Disk Read Interrupt
        JC      ERROR_EXIT_2        ; Quit if there was an error
        POPF                        ; Restore the user flags
        POP     DS                  ; Restore DS
.
. Free the allocated memmory prior to returning to DOS
.
        MOV     AH,49H              ; DOS FUNCTION
        INT     21H                 ; Function Request
        RET                         ; Return to DOS
;
; Error return without popping the stack
;
ERROR_EXIT_1:
        RET                         ; Return to DOS
;
; Error return with popping the stack
;
ERROR_EXIT_2:
        POP     DI                  . Throw away the user flags
        POP     DS                  . Restore DS
.
; Free the allocated memory prior to returning to DOS
;
        MOV     AH,49H              ; DOS Function
        INT     21H                 ; Function Request
        RET                         ; Return to DOS
.
END_OF_PROG     EQU     $
;
CODESEG ENDS
;
; ****************************************************************
;
        END     BEGIN
```

新產品介紹

簡易資料庫管理系統RAPIDFILE

ASHTON TATE 公司（以 dBASE II 馳名）最近推出了一個簡單易用的資料庫軟件 —— RAPIDFILE，對象爲沒有時間自行編寫程式或不需太多功能的用戶。RAPIDFILE 可用作管理資料、產生報告、塡寫固定格式信件、印製 LABEL 等，足以應付一般的需要。RAPIDFILE 包括了與 dBASE 相似的拉下（PULL-DOWN）目錄，以及與 1-2-3 相似的介面和指令結構。顯示／編印資料有不同格式可供選擇，輸入之時容許五種不同的欄，包括文字、數字、報告、日期及計算值。另一方面，RAPIDFILE 可使用 dBASE III 的檔案。■

圖：RAPIDFILE 容許不同的資料顯示形式



產品名稱：RAPIDFILE
售　　價：$395 (U.S.)
硬件要求：TWO DISK DRIVES, 256K RAM
生 產 商：ASHTON TATE, 20101 HAMILTON AVE., TORANCE, CA 90502.

最有效軟/硬碟機檢查工具

DISKSCAN磁碟機維護系統

每次開動PC之時，ROM內的BIOS都會替系統內裝設的記憶進行測試。這測試提供一個保護，防備因記憶的錯誤而破壞隨後的工作。

不幸地，BIOS並沒有對系統內的硬磁碟履行相似的測試，那麼硬磁碟機可能出現壞磁區(SECTOR)而閣下却一無所知，直至"Abort，Retry，Ignore"訊息打出才知道DOS再不能讀取一個檔案。

本文程式DISKSCAN提供一簡便方法來監督着硬磁碟的狀態，以避免上述問題。它讀取磁碟每一個磁區，然後報告所有找到的錯誤。初時，DISKSCAN每次讀取64K(多數磁碟是每磁區512 BYTE，即共128個磁區)，假如當程式讀取這組磁區時接觸到一個錯誤，它會返回第一個磁區並續個讀取，報告那一個是壞磁區及故障的情形。

DISKSCAN會工作於任何磁碟或其它BLOCK裝置。包括軟磁碟、硬磁碟、碟帶機，及RAMDISK。

倘若沒有發現任何壞磁區，DISKSCAN需時約兩分半鐘測試一個標準IBM 10 MB容量硬磁碟機。當然，越大記憶的磁碟需時越久，不單止因爲它們龐大數目的磁區，亦因爲錯誤報告的次數同時提高，儘管不必將DISKSCAN放入AUTOEXEC·BAT檔案內，定時使用它可在資料消失前作出備份。

筆者除討論DISKSCAN如何工作外，還會爲各位一起探索幾類有關的東西—磁碟的組織，與及程式如何能夠由DOS找出這資料。

檢查硬碟機

只要輸入DISKSCAN [drive:]就可以執行程式，倘若沒有提供選擇性的參數，程式會使用預設磁碟機。當它執行時，DISKSCAN不斷顯示它現時正在讀取之磁區範圍。因爲大部份磁碟使用512-BYTES長度的磁區，DISKSCAN通常一次讀取128個磁區。如果它遇到一個錯誤，它對這組磁區顯示"Error"，之後跳往下一行，並開始從這些磁區每次讀取一個。在這單獨磁區讀取過程中遇到的每個錯誤，DISKSCAN把磁區數字及錯誤類型兩者顯示出來。

程式鍵入方法

閣下可以鍵入列表一之程式來創造DISKSCAN·ASM檔案，以MASM組合，以LINK連結，忽略"no stack segment"信息，並以EXE 2BIN把它轉換爲·COM格式。警告：倘若閣下決定鍵入組合語言列表，請小心，不要意外地把ReadSectors內的

Int 25h

讀取磁碟改爲

Int 26h

中斷向量(Interrupt)26h是寫入磁碟，倘若用它代替正確的插斷25h，結果是把一大堆垃圾通過DISKSCAN寫入整張磁碟。嘗試執行程式前請重覆檢查這行的碼。對硬磁碟執行程式前，首先用在沒有資料的磁碟上，測試程式是否無誤。

讀取磁碟

自行編寫程式時，通常以檔案形式工作，並且不會理會DOS如何把那些檔案貯存於磁碟內。事實上，DOS的主要工作是以操縱目錄，檔案分配表(FILE ALLOCATION TABLE以下稱FAT)，及資料區來將檔案組織於磁碟內，因此閣下無需理會，而DISKSCAN則是一個需要直接存取磁碟磁區的程式。

BIOS提供中斷13h作爲直接讀取或寫入磁碟磁區的用途。然而DISKSCAN使用DOS提供的另兩個直接存取磁碟磁區之中斷(25h是讀取而26h是寫入)。

PC-DOS程式CHKDSK，RECOVER，及DEBUG全部都是直接存取磁碟磁區，同時全部使用中斷25h及26h，而非13h。(FDISK，SYS，DISKCOPY，及DISKCOMP使用13h，不過FDISK之所以採用是因爲DOS辨別磁碟前閣下已經使用它。FORMAT及DISKCOPY一定要格式化一隻磁碟，因而唯有通過中斷13h完成）。

避免使用插斷13h的首要原因是並非所有外置記憶都支持它。目前，很多在CONFIG·SYS檔案內設定的BLOCK裝置(例如RAMDISK，磁帶備份系統等）並非全部使用中斷13h，中斷13h實際上只適用於IBM磁碟機或相容產品而已。

中斷25h及26h比起中斷13h亦較易使用，使用時無需知道磁碟結構規定(磁軌、磁區、磁鼓、磁面)，同時閣下可以由零開始，簡單地當此磁碟爲連成一串的連續磁區。

DOS的限制

DISKSCAN 磁碟機維護系統

使用中斷25h及26h的唯一眞正缺點是它們只可存取硬磁碟的DOS使用部份，而並非是整張磁碟。整張磁碟都是劃分給DOS(大都如此)，插斷25h及26h仍然不能夠取出存於硬磁碟第一個磁鼓第一面的第一磁區之劃分表。對插斷25h及26h，磁區零是磁碟上下一個物質磁區，亦即包含DOS的BOOT記錄之磁區。

當應用中斷25h及26h於程式內時，要留意它們會與堆疊區混雜。當DOS由這些中斷返回時，它使用Carry旗號標誌錯誤，然而原本的旗號仍然存在堆疊區。因而，閣下的程式必定要從堆疊區取消旗號，否則程式的運作可能產生問題。儘管在組合語言內只是小意思，這問題可防碍閣下以中斷25h及26h使用組合語言(例如Turbo Pascal intr或Microsoft C's int86)內的內建插斷功能。最好是編寫一個組合語言副程式代替。

磁碟格式

閣下或會由DOS技術參考手冊中獲悉，一張磁碟被劃分爲四個分開的區域：BOOT磁區(通常只佔第一個磁區)，檔案分配表(FAT通常一張磁碟維持兩個相同備份)，目錄，及資料。

在DOS 1.1版本中，決定磁碟內的區域每一個使用多少磁區數目非常容易，因爲兩個支持磁碟格式化(雙面及單面)的格式已被固定。不過假如閣下以FDISK或VDISK 3.x將硬磁碟分爲幾個部份，你將看到DOS 2.0及更新版本可以支持差不多所有的磁碟格式。

用DISKSCAN觀察硬磁碟所有磁區，必定要知道磁碟或(硬磁碟情況下)DOS區分內磁區的總數，這並非輕微的問題，原因是這些資料並不能簡單地由DOS提供的功能呼喚中獲得。只可以DOS的中斷21h功能呼喚1Bh，1Ch，及36h來取得。這些回覆的CX值等於每個磁區的位元組數目，DX等於一組的數目，和(僅36h)BX等於空組別的數目。然而這些組別(它們由一個或多個磁區組成)只屬於資料而已。這些功能呼喚不會告訴閣下由FAT或目錄所使用的磁區數目。

因此，應該用DOS中斷21h功能呼喚計算出磁碟上的磁區總數。DOS在系統內爲每一隻磁碟維持一個磁碟作業系統磁碟區(DOS DISK BLOCK以下稱DDB)，DDB描述磁碟的格局，而功能呼喚32h可讓閣下觀看它。因爲IBM明顯地在DOS技術參考手册內沒有爲DDB及功能呼喚32h提供說明文件，筆者特別在表一詳列它們的用途。

要完成表一之數據，有關之功能呼喚1Fh回覆與32h相同的資料，然而只供預設之磁碟機（雖然資料有些微差異，這功能呼喚可工作於DOS 1.1)。例如表一的每一個DDB包含一個指向下一個好double-word (D Word)位址，DDB就是如此在DOS內組成一條鏈一樣。這條鏈以磁碟機A：開始，並且在系統內以字母順序排列(或其他BLOCK裝置)。呼喚52h回覆ES：BX指向DWord，表示鏈的開頭；即是，存於ES：BX的DWord是給A：機之DDB位址。倘若中間有變動，功能呼喚1Fh及32h會引致磁碟存取發生測驗。功能呼喚52h自己並不會引致磁碟存取，因此轉換磁碟而却未曾以它存取則它回覆的資料可能有錯。

表一：中斷 32h 的功能表

```
DOS Int 21h Function Call 32h -- Get "DOS Disk Block"
------------------------------------------------------------

On Entry        Register Contents
---------       -----------------
   AH           32h
   DL           Number of Disk Drive (0=default, 1=A, etc.)

On Return       Register Contents
---------       -----------------
   AL           = 00 if drive exists, FFh for invalid drive
 DS:BX          Pointer to DOS Disk Block

Description
-----------
The DOS Disk Block contains the following information at
the indicated offsets of DS:BX

Offset     Type     Data
------     ----     ----
  00       Byte     Drive: 0=A, 1=B, etc.
  01       Byte     Unit within driver (0, 1, 2, etc.)
  02       Word     Bytes per sector
  04       Byte     Sectors per cluste - 1
  05       Byte     Cluster to sector shift
  06       Word     Number of reserved (boot) sectors
  08       Byte     Number of File Allocation Tables
  09       Word     Number of root directory entries
  0B       Word     Sector number of cluster 2
                      (cluster 2 is first data cluster)
  0D       Word     Number of clusters + 1
                      (or last cluster number)
  0F       Byte     Sectors for FAT
  10       Word     Sector number of directory
  12       DWord    Address of device header
  16       Byte     Media Descriptor Byte
  17       Byte     Zero if disk has been accessed
  18       DWord    Address of next DOS Disk Block
                      (FFFF if last one in chain)
```

DOS的BIOS參數區呼叫磁碟機裝置計算出DDB內的資料(BIOS參數區在DOS技術手册內有詳細說明)。DDB包含著磁碟機裝置標題的位址，用這個位址，包括一個BIOS參數區呼喚，一個程式且能夠直接使用磁碟機裝置。閣下或會對DOS功能呼喚53h感興趣，它將BIOS參數區(由DS：SI指示)列表之格式內的資料翻譯爲DDB列表(由ES：BP指出)之格式。

DISKSCAN僅需由DDB取得幾個資料。規格上，它需要應用於資料的組別數目，每組的磁區數目，那裡是資料開始之磁區數碼，及每磁區的BYTES數目(一隻磁碟最普遍採用每磁區512 BYTES，然而RAMDISKS會較少，而超過32 MB容量的磁碟會較多)，其他程式(例如CHKDSK)通常由DDB要求更多資料。

無論何時，在目錄或FAT內直接存取磁碟磁區時，不要胡亂假設那裡才是這些磁區，而應該用功能呼喚32h來找出。

不斷嘗試

當程式利用DOS功能呼喚存取檔案時，DOS會呼叫磁碟機裝置讀取及寫入適當的磁區。磁碟機裝置於是使用中斷13h (IBM或兼容之硬磁碟或軟磁碟情況下)、直接去到裝置的輸入/輸出埠，或者只存取記憶(RAMDISK)。

倘若磁碟機裝置因磁碟出現問題而不能履行要求，DOS再次嘗試呼叫磁碟機裝置。假如經三次嘗試後磁碟機裝置仍回覆錯誤，DOS便顯示它的錯誤訊息 "Abort，Retry，Ign

ore"。

不過中斷25h及26h只對磁碟機裝置進行單獨一次的直接呼喚。這些中斷在遇到錯誤發生時回覆一錯誤碼，取代導致"Abort，Retry，Ignore"訊息。倘若閣下由DISKSCAN獲悉"Abort，Retry，Ighore"，它只可能由32h產生。表示磁碟機裝置之介質也不能辨別。以軟磁碟來說，可說是門仍開著或磁碟未被格式化。對硬磁碟來說，已被FDISK及FORMAT指令格式而由功能呼喚32h獲得錯誤便是問題嚴重。

雖然中斷25h及26h只呼喚磁碟機裝置僅一次而不再重覆嘗試，有時閣下或會留意到(特別是當DISKSCAN用於軟磁碟時)某些東西在DISKSCAN報告錯誤前存取磁碟五次。IBM為了提高磁碟機裝置的速度，把存取磁碟時設定磁軌追踪時間爲零，這樣會引致很多誤讀產生，因此磁碟機裝置在放棄及回覆錯誤前應嘗試五次。

追踪壞磁區

當初次格色化你的硬磁碟時，你或會留意到FORMAT指令指出磁碟有些壞磁區(大多如此)。當閣下執行CHKDSK，或會看到一行告訴閣下磁碟一些BYTES是在壞磁區內。此外，儘管CHKDSK可能指出目前你的PC—XT硬磁碟在壞磁區有4096 BYTES，DISKSCAN只告訴只有一個512—BYTE的磁區壞了。這是因爲FAT以組別工作。假如任何一個組別的磁區壞了，整個組別一定被影响。

幸運地，當FORMAT指令遇到壞磁區時，它在FAT內放下標記，而這些磁區不能供資料使用。DISKSCAN對應它找到的壞磁區，與在格式時DOS放入FAT的列表對比，當適當時回覆再報告"Flagged as bad'"。

最佳的造法是格式化硬磁碟後立刻執行DISKSCAN，並把它接觸到的壞磁區印製成列表。因而，當閣下日後執行DISKSCAN，可用新的列表與舊的比較。

錯誤訊息

雖然DISKSCAN有錯誤訊息供它能夠找到的13種不同錯誤類型，它們當中有些明顯地沒有應用需要。閣下最普通看到的錯誤類型如下：

●CRC Error：當資料被存入磁碟時，一核對值(名爲Cyclic Redundancy Check)亦被存入磁碟。倘若當資料被讀取時，這數值與重新計算之值不相符，則假設資料無效，閣下便會得到這訊息。

●Sector Not Found：指出用FORMAT創造時標誌著之磁區範圍已被破壞，同時正讀取之磁區不再清楚地記錄於磁軌上。閣下或會於一列內得到幾個這樣的訊息。

●Read Fault and General Failure：這些是一般錯誤並不會落入其它範畴。

DISKSCAN的第二組訊息對閣下偵錯可能會有較大的幫助。

●File Alloc Table and Can't Read FAT：這指出非常嚴重的情況，宜立刻停止任何操作，並使用DOS的COPY指令盡可能把資料備份，用Restore指令恢復備份檔案前重新把硬磁碟格式化。

●Boot Sector：這問題較易解決，原因是閣下可隨時由A：機重新BOOT碟。放DOS磁碟入A：機並執行SYS C:指令，之後鍵入COPY COMMAND．COM C：取代。

●Root Directory：此處之錯誤能夠防止閣下隨後載入檔案或貯存更新檔案之能力。倘若執行CHKDSK可能會得到有關於未被分配的組別鏈或交叉鏈檔案的訊息，可執行CHKDSK/F把它們固定。

●Unallocated：磁碟發生磁區未被分配，錯誤並不嚴重，原因在於此磁區目前並未被任何檔案使用。另一方面，當一檔案意外地寫進此區域時，閣下會得到"Abort，Retry，Ignore."訊息。當此情形發生時，選擇"Ignore，"把剛貯存之檔案改名，再次貯存你的檔案，之後在第一個檔案使用RECOVER。這方法令DOS把壞組別設立旗號及不會於日後再嘗試使用。跟著把第一個檔案刪除，保留第二個貯存版本。

●Use by file：那一個檔案——及甚麼能夠被它們貯存一是當閣下得到這訊息的疑問。

不幸地，把已很長的DISKSCAN碼擴大，從而確認那些檔案包含著這些壞磁區在本文是行不通的。DISKSCAN僅會開始存取目錄及副目錄登記，找出那裡是每一檔案開頭之組別，之後觀看FAT組別鏈直至同一位置再出現。

RECOVER的使用方法

一檔案如有壞磁區於它的中央時，需要用到這檔案的程式可能出現問題。首要工作是以另一名稱抄至另一檔案，當DOS詢問"Abort，Re-try，Ignore."時選擇"R"。倘若沒法工作，閣下便有機會使用一個所有DOS指令中最被忽略的RECOVER指令。使用時鍵入

RECOVER filename

這裡"filename"是含有壞磁區之檔名。RECOVER會做兩件事：首先，它會標記壞磁區（更爲精確地，它標記那些含有壞磁區的組別）如同FAT內之標記，因此它們不能再被使用。其次，RECOVER逸去那些壞磁區把檔案拼湊一起。

（然而，在沒有檔名參數的情況下不要使用RECOVER：它會以不同的方法執行及不會如預期地工作）。

RECOVER後的檔案是否有用？這取決於，如它是一．COM或．EXE程式，倘若你嘗試執行它便可能失敗，因爲它失去部份資料。如果它是一個Worksheet或database檔案，程式便會出現混亂。倘若是文字檔或原程式列表，閣下就較幸運。它雖然損失部份資料，但大部份仍然存在，閣下或會有能力把其餘資料重新組織。

將已經顯現壞磁區的磁碟或資料重新恢復雖然可用RECOVERY指令直接補救，然而，如有可能，閣下定必希望完全避免這過程。

因此DISKSCAN的最有用功能是在閣下的硬磁碟顯示衰老迹象時及早爲閣下提出警告。倘若閣下規律地爲硬磁碟備份及執行DISKSCAN，至少可防備磁碟大災難及再向你的資料安全作保證。■

列表一：DISKSCAN.ASM 組合語言原程式

```
; DISKSCAN.ASM -- Checks out disk by reading sectors
; ---------------------------------------------------

CSEG                    Segment
          Assume        CS:CSEG, DS:CSEG, ES:CSEG, SS:CSEG
          Org           100h
Entry:                  Jmp     Begin

; All Data
; --------
                    db          ' Copyright 1986 COMPULINK MAGAZINE
          db        ' Programmed by COMPUHOUSE         '
DriveError          db          'Invalid Drive$'
DosVersErr          db          'Needs DOS 2.0+$'
MemoryError         db          'Needs 64K$'
ReadSegment         dw          ?
DriveNum  db        ?
DiskBlock db        18 dup (?)
TotalSectors        dw          ?
SectorsIn64K        dw          ?
StartSector         dw          0
SectorLabel2        db          9,'Sector $'
SectorLabel         db          13,'Sectors $'
DashLabel db        ' - $'
ErrorLabel          db          ': Error!'
CRLF                db          13,10,'$'
ErrorAddr dw        Err0,Err1,Err2,Err3,Err4,Err5,Err6,Err7
          dw        Err8,Err9,ErrA,ErrB,ErrC,ErrD,ErrD,ErrD
Err0                db          'Write Protect$'
Err1                db          'Unknown Unit$'
Err2                db          'Drive Not Ready$'
Err3                db          'Unknown Command$'
Err4                db          'CRC Error$'
Err5                db          'Request Length$'
Err6                db          'Seek Error$'
Err7                db          'Unknown Media$'
Err8                db          'Sector Not Found$'
Err9                db          'No Paper$'
ErrA                db          'Write Fault$'
ErrB                db          'Read Fault$'
ErrC                db          'General Failure$'
ErrD                db          'Undocumented Error$'
BootSectMsg         db          'Boot Sector$'
RootDirMsg          db          'Root Directory$'
BadFatMsg db        'File Alloc. Table$'
InUseMsg  db        'Used by file$'
NotInUseMsg         db          'Unallocated$'
BadFlagMsg          db          'Flagged as bad$'
FatReadMsg          db          "Can't Read FAT$"
Divisors  dw        10000, 1000, 100, 10, 1 ; For decimal conversion

; Check Drive Parameter, DOS Version, and Enough Memory
; ------------------------------------------------------

ErrorExit:          Mov     AH,9                ; Write error message
          Int       21h                 ;       through DOS
          Int       20h                 ; And terminate

Begin:              Mov     DX, Offset DriveError   ; Possible message
          Or        AL, AL              ; Check Drive Validity Byte
          Jnz       ErrorExit           ; If not zero, invalid drive
          Mov       DX, Offset DosVersErr   ; Possible message
          Mov       AH, 30h
          Int       21h                 ; Get DOS Version Number
          Cmp       AL, 2               ; Check for 2.0 or later
          Jb        ErrorExit           ; If not, terminate with message
          Mov       DX, Offset MemoryError  ; Possible error message
          Mov       BX, 256+Offset EndProg  ; Set beyond program
          Mov       SP, BX              ; Move stack closer to code
```

```
            Add     BX, 15          ; Add 15 to round up
            Mov     CL, 4           ; Divide BX by 16
            Shr     BX, CL
            Mov     AH, 4Ah         ; Free allocated memory
            Int     21h             ;   by calling DOS Set Block
            Jc      ErrorExit       ; Terminate on error
            Mov     BX, 1000h       ; Ask for 64K bytes
            Mov     AH, 48h         ;   by using DOS
            Int     21h             ;   Allocate Memory call
            Jc      ErrorExit       ; Terminate on error
            Mov     [ReadSegment], AX       ; Save segment of memory block

; Get Disk Information From DOS
; -----------------------------

            Mov     DL, DS:[005Ch]  ; Get Drive Parameter
            Push    DS              ; Save DS
            Mov     AH, 32h         ; Call DOS to
            Int     21h             ;   get DOS Disk Block (DS:BX)
            Mov     SI, BX          ; Now DS:SI points to Disk Block
            Mov     DI, Offset DiskBlock    ; DI points to destination
            Mov     CX, 18          ; 18 bytes to copy'
            Cld                     ; Forward direction
            Rep     Movsb           ; Move 'em in
            Pop     DS              ; Get back DS
            Mov     BX, Offset DiskBlock    ; BX to address Disk Block
            Mov     DX, 1           ; Set DX:AX to 65,536
            Sub     AX, AX
            Div     Word Ptr [BX + 2]       ; Divide by Bytes Per Sector
            Mov     [SectorsIn64K], AX      ; Save that values
            Mov     AX, [BX + 13]           ; Last Cluster Number
            Dec     AX                      ; AX = Number of Clusters
            Mov     CL, [BX + 5]            ; Cluster to Sector Shift
            Shl     AX, CL                  ; AX = Number Data Sectors
            Add     AX, [BX + 11]           ; Add First Data Sector
            Mov     [TotalSectors], AX      ; AX = Number Total Sectors
            Mov     AL, DS:[005Ch]  ; Drive Number (0=def, 1=A)
            Dec     AL              ; Make it 0=A, 1=B
            Jns     GotDriveNumber  ; If no sign, not default drive
            Mov     AH, 19h         ; Get current disk
            Int     21h             ;   by calling DOS

GotDriveNumber:     Mov     [DriveNum], AL  ; Save Drive Number (0=A, 1=B)

; Start Reading
; -------------

MainLoop:   Mov     DX, Offset SectorLabel  ; String to display on screen
            Call    StringWrite             ; Display it
            Mov     AX, [StartSector]       ; Starting sector number
            Call    WordWrite               ; Display number on screen
            Mov     DX, Offset DashLabel    ; String containing a dash
            Call    StringWrite             ; Display it on the screen
            Mov     CX, [SectorsIn64K]      ; Number of sectors to read
            Add     AX, CX                  ; Add it to starting sector
            Jc      NumRecalc
            Cmp     AX, [TotalSectors]      ; See if bigger than total
            Jbe     NumSectorsOK            ; If so, proceed

NumRecalc:          Mov     AX, [TotalSectors]      ; Otherwise get total sectors
            Mov     CX, AX                  ; Move it to CX also
            Sub     CX, [StartSector]       ; Now CX = sectors to read

NumSectorsOK:       Dec     AX                      ; AX = last sector to read
            Call    WordWrite               ; Display it on screen
            Call    ReadSectors             ; Read the sectors
            Jnc     NextSectors             ; If no error, skip detail
            Call    ReadSectors             ; Repeat read
            Jnc     NextSectors             ; If still no error, skip

DiskError:          Mov     DX, Offset ErrorLabel   ; String saying "Error!"
            Call    StringWrite             ; Display it on screen

ErrorLoop:          Push    CX                      ; Now save previous number
            Mov     CX, 1                   ; So we can read one at a time
            Call    ReadSectors             ; Read one sector
            Jnc     NoError                 ; If no error, proceed
            Mov     BL, AL                  ; Save error code
```

```
            Mov     DX, Offset SectorLabel2 ; String with "Sector "
            Call    StringWrite             ; Display it on screen
            Mov     AX, [StartSector]       ; The sector we just read
            Call    WordWrite               ; Display it on screen
            Mov     DX, Offset DashLabel    ; String with a dash
            Call    StringWrite             ; Display it on screen
            And     BL, 0Fh                 ; Blank out error top bits
            Sub     BH, BH                  ; Now BX is error code
            Add     BX, BX                  ; Double it for word access
            Mov     DX, [ErrorAddr + BX]    ; Get address of message
            Call    StringWrite             ; Display message on screen
            Call    FindSector              ; See where sector is
            Mov     DX, Offset CRLF         ; String for new line
            Call    StringWrite             ; Do carriage ret & line feed

NoError:    Inc     [StartSector]           ; Kick up the start sector
            Pop     CX                      ; Get back counter
            Loop    ErrorLoop               ; And read next sector
            Mov     AX, [StartSector]       ; Sector of next group
            Jmp     Short CheckFinish       ; Check if at end yet

NextSectors:        Mov     AX, [StartSector]       ; For no error, increment
            Add     AX, [SectorsIn64K]      ;   StartSector for next group
            Jc      Terminate               ; (If overflow, terminate)
            Mov     [StartSector], AX       ; And save it

CheckFinish:        Cmp     AX, [TotalSectors]      ; See if at then end
            Jae     Terminate               ; If so, just terminate
            Jmp     MainLoop                ; If not, do it again

Terminate:          Int     20h                     ; Terminate

; Find Sector in FAT to see if used by file, etc.
; ----------------------------------------------

FindSector:         Mov     DX, Offset DashLabel            ; Print dash
            Call    StringWrite
            Mov     AX, [StartSector]               ; Sector with error
            Mov     DX, Offset BootSectMsg          ; Set up message
            Cmp     AX, Word Ptr [DiskBlock + 6]    ; See if sector boot
            Jb      PrintMsg                        ; If so, print as such
            Mov     DX, Offset BadFatMsg            ; Set up message
            Cmp     AX, Word Ptr [DiskBlock + 16]   ; See if sector in FAT
            Jb      PrintMsg                        ; If so, print as such
            Mov     DX, Offset RootDirMsg           ; Set up message
            Cmp     AX, Word Ptr [DiskBlock + 11]   ; See if sector in dir
            Jb      PrintMsg                        ; If so, print as such
            Push    [StartSector]                   ; Save the sector
            Mov     AX, Word Ptr [DiskBlock + 6]    ; Reserved sectors
            Mov     [StartSector], AX               ; Start of first FAT
            Mov     CL, [DiskBlock + 15]            ; Sectors for FAT
            Sub     CH, CH                          ; Zero out top byte
            Call    ReadSectors                     ; Read in FAT
            Pop     [StartSector]                   ; Get back bad sector
            Mov     DX, Offset FatReadMsg           ; Set up possible msg
            Jc      PrintMsg                        ; If read error, print
            Mov     AX, [StartSector]               ; Get bad sector
            Sub     AX, Word Ptr [DiskBlock + 11]   ; Subtract data start
            Mov     CL, [DiskBlock + 5]             ; Sector Shift
            Shr     AX, CL                          ; Shift the sector
            Add     AX, 2                           ; AX is now cluster
            Push    ES                      ; Save ES for awhile
            Mov     ES, [ReadSegment]       ; ES segment of FAT
            Cmp     Word Ptr [DiskBlock + 13], 0FF0h; 12 or 16-bit FAT?
            Jge     Fat16Bit                ; And jump accordingly
            Mov     BX, AX                  ; This is cluster number
            Mov     SI, AX                  ; So is this
            Shr     BX, 1                   ; This is one-half cluster
            Mov     AX, ES:[BX + SI]        ; BX + SI = 1.5 CX
            Jnc     NoShift                 ; If no CY from shift, got it
            Mov     CL, 4                   ; If CY from shift must
            Shr     AX, CL                  ;   shift word 4 bits right

NoShift:    Or      AX, 0F000h              ; Now put 1's in top bits
            Cmp     AX, 0F000h              ; See if zero otherwise
            Jmp     Short CheckWord         ; And continue checking

Fat16Bit:   Mov     BX, AX                  ; This is cluster number
```

```
            Shl     BX, 1                       ; Double it
            Mov     AX, ES:[BX]                 ; Pull out word from sector
            Or      AX, AX                      ; See if zero (unallocated)

CheckWord:          Pop     ES                          ; Get back ES
            Mov     DX, Offset NotInUseMsg      ; Set up possible message
            Jz      PrintMsg                    ; If so, print message
            Mov     DX, Offset BadFlagMsg       ; Set up possible message
            Cmp     AX, 0FFF7h                  ; See if cluster flagged bad
            Jz      PrintMsg                    ; If so, print message
            Mov     DX, Offset InUseMsg         ; If not, cluster is in use

PrintMsg:   Call    StringWrite                 ; Print cluster disposition
            Ret                                 ; And return

; Read Sectors (CX = Number of Sectors, Return CY and AL for error)
; -----------------------------------------------------------------

ReadSectors:        Push    BX                          ; Push all needed registers
            Push    CX
            Push    DX
            Push    DS
            Mov     AL, [DriveNum]              ; Get the drive number code
            Sub     BX, BX                      ; Buffer address offset
            Mov     DX, [StartSector]           ; Starting Sector
            Mov     DS, [ReadSegment]           ; Buffer address segment
            Int     25h                         ; Absolute Disk Read
            Pop     BX                          ; Fix up stack
            Pop     DS                          ; Get back registers
            Pop     DX
            Pop     CX
            Pop     BX
            Ret                                 ; Return to program

; Screen Display Routines
; ----------------------

WordWrite:          Push    AX                          ; Push some registers
            Push    BX                          ; AX contains word to display
            Push    CX
            Push    DX
            Push    SI
            Mov     SI, Offset Divisors         ; SI points to divisors
            Mov     CX, 4                       ; CL counter; CH zero blanker

WordWriteLoop:      Mov     BX, [SI]                    ; Get divisor
            Add     SI, 2                       ; Increment SI for next one
            Sub     DX, DX                      ; Prepare for division
            Div     BX                          ; Divide DX:AX by BX
            Push    DX                          ; Save remainder
            Or      CH, AL                      ; See if zero
            Jz      LeadZero                    ; If so, do not display it
            Add     AL, '0'                     ; Convert number to ASCII
            Mov     DL, AL                      ; Print out character
            Mov     AH, 2                       ;   by calling DOS
            Int     21h

LeadZero:   Pop     AX                          ; Get back remainder
            Dec     CL                          ; Decrement counter
            Jg      WordWriteLoop               ; If CL still > 0, do it again
            Mov     CH, 1                       ; No more zero blanking
            Jz      WordWriteLoop               ; Convert last digit to ASCII
            Pop     SI                          ; Get back pushed registers
            Pop     DX
            Pop     CX
            Pop     BX
            Pop     AX
            Ret

StringWrite:        Push    AX                          ; Displays string from DX
            Mov     AH, 9                       ;   to screen by calling DOS
            INT 21H
  POP AX
  RET

ENDPROG LABEL BYTE              ; END OF PROGRAM
CSEG    ENDS
        END     ENTRY ■
```





電腦時代產品總目錄・歡迎選購



蘋果中文咭應用程式專集



MOCKING BOARD 技術應用



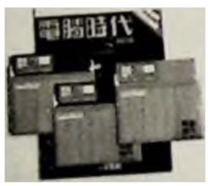
GPLE 使用方法詳析



128K RAM
咭應用技術詳析



電腦時代精裝合訂本（13-24期）



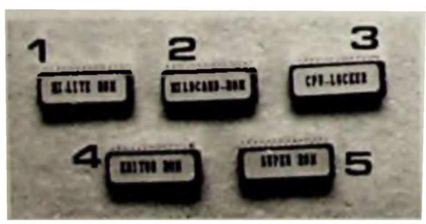
電腦時代精裝合訂本（1—12期精選程式集）連三張程式磁碟



128K RAM 咭應用程式專集（第一輯）



DUAL ROM CARD



1. 80 COL HI-LITE ROM ($20)
2. WILDCARD ROM (COMMING SOON)
3. CPU LOCKER
4. EDITOR ROM ($15)
5. SUPER ROM ($35)



KEYBOARD LOCKER

請與我們一起
走在潮流的尖端

3-880580

COMPUNET 資訊網絡中心

未來流行的時尚——電腦通訊

用最低廉的代價・獲取無限的資源

服務範圍：提供免費公開版權之軟件存取。學術經驗交流。電子郵遞。與及任何想像力能達到的範疇。

服務對象：任何擁有電腦的人士。



蘋果電腦用家必備之電腦雜誌

新版電腦時代 1－12期

精裝程式合訂本

隨書附同一套三張程式磁碟

[illegible]至 12 期精裝合訂本出版後，受到廣泛而熱烈的歡迎，很快便已銷售一空。許多讀者前來出版社[illegible]，我們只能很遺憾的告訴他們 —— 我們連一本存書也沒有。近月來我們接獲許多讀者要求重印[illegible]讀者失望，出版社決定再重新印行這本合訂本。

[illegible]電腦時代第 1 至 12 期所刊出的程式文章，去蕪存菁地精選出超過六十個精彩而實用的軟件程式。[illegible]多個程式無論任何時刻都是歷久常新，適合新入行的電腦用家或經已是我們長期擁護的讀者。

新合訂本比以前所出版的合訂本更超值，因為它是連同三張精美的程式磁碟一併出版，換句說話，你只要以一百元的代價，就獲得價值超過 475 元的東西。新版精裝程式合訂本刊登的六十多個程式：

電腦時代月曆
COMPUTEXT文字處理系統
雜誌期刊文章及目錄頁數索引紀錄系統
自製磁碟索引封套器
螢幕健美操——能令神經鬆弛的程式
私人約會備忘日曆
會講廣東話的蘋果電腦
電子郵箱
磁碟檔案索引黃頁分類系統
可隨意訂定格式之資料紀錄處理系統
MARCO全能功能鍵盤
家庭購物備忘及食物日用品存貨管理系統
家庭開支及個人消費管理系統
任意設計目錄索引顯示程式
將工具程式放入DOS佔用之三條磁軌上
最完整的HELLO程式：GO II
將DOS 3.3移上16K RAM卡
DOS 3.3 指令修改器
自動減省程式長度 加快執行效率程式
十六進機械碼編印格式
RAM卡控制中心
DOS殺手：令磁碟增加三條磁軌儲存空間的工具
[illegible]存入RAM卡
[illegible]文字頁存入RAM卡
將RAM卡作為磁碟機使用
有多項功能的AMPERSAND工具程式
[illegible]化程式列表格式編印器
[illegible]
[illegible]
[illegible]

英文語句文法文析練習系統
初中數學套件之(一)
初中數學套件之(二)
初中代數練習
符號逆轉型(RPN)計數機
[illegible]
[illegible]不滅定律
數字認識初階
[illegible]
四則運算訓練
加數計算機
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
立體迷宮
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
JACKIES HEAD
[illegible]
[illegible]

特約經銷處

- 電腦時代九龍服務中心
- 中環三聯書店（中環街市對面）
- 北角教育人員專業協會
- 深水埗黃金商場地庫21號萬達電腦公司
- 西環屈地街 22 號地下創業雜誌中心